UNIVERSITY OF
ILLINOIS LIBRARY
AT URBANA-CHAMPAIGN
ENGINEERING

NOTICE: **Return or renew all Library Materials! The *Minimum* Fee for each Lost Book is $50.00.**

JUN 27 1988

The person charging this material is responsible for its return to the library from which it was withdrawn on or before the **Latest Date** stamped below.

Theft, mutilation, and underlining of books are reasons for disciplinary action and may result in dismissal from the University.
To renew call Telephone Center, 333-8400

UNIVERSITY OF ILLINOIS LIBRARY AT URBANA-CHAMPAIGN

L161—O-1096

Digitized by the Internet Archive
in 2012 with funding from
University of Illinois Urbana-Champaign

http://archive.org/details/programmingmanua148behr

.84
63c
148-C

Engin

Center for Advanced Computation

UNIVERSITY OF ILLINOIS AT URBANA-CHAMPAIGN

URBANA, ILLINOIS 61801

CAC Document No. 148-C

PROGRAMMING MANUAL
Computer Aided Dispatch System for the Police Departments of Oak Park, River Forest, and Forest Park, Illinois

by
William Behr and Warner Brigham

March 1, 1974

PROGRAMMING MANUAL
(Part I)

Computer Aided Dispatch System
for the
Police Departments of Oak Park,
River Forest, and Forest Park, Illinois

by

William Behr and Warner Brigham

Center for Advanced Computation
University of Illinois at Urbana-Champaign
Urbana, Illinois 61801

March 1, 1974

This work was supported by a grant from the Illinois Law Enforcement Commission to the Village of Oak Park through an Agreement for Cooperative Investigation between Oak Park and the Board of Trustees of the University of Illinois.

© 1973 by the Board of Trustees of the University of Illinois

FOREWORD

The following report gives a description and the detailed line code for each of the software modules required to make the Automated Interactive Dispatch System operable. This Automated Interactive Dispatch Software System is composed of many software modules which interact with each other mainly through the system vector tables and terminal information tables. The hardware required for the software module to operate is on a Data General Nova 820 computer, a Century #114 disc pack system, and a Decision disc controller. They operate under DINOS (Decision Incorporated Nova Operating System) and DDOS (Decision Disc Operating System). It is essential that these operating systems are functional on the system before the Automated Interactive Dispatch Software System will operate.

The reader of these reports should be familiar with the System Manual of the Computer Aided Dispatch System before looking at this Programming Manual. He should also be familiar with the Data General Nova Computers, their assembly language coding procedure, and somewhat familiar with the two operating systems from Decision, Inc. Each module has a short description preceding the code and the detailed code has comments on each line of code describing exactly each line of code.

SYSTEM VECTOR TABLE (SVT)

The system vector table defines all system vectors and parameters relevant to the system as a whole.

```
PAGE  1        A I D S     S Y S T E M     V E C T O R     T A B L E
02/20/74
                .HEAD   A I D S     S Y S T E M     V E C T O R     T A B L
                .TITLE  SVT
                .EXTN   DUMP.,EDIT.     ; FORMAT DISPLAY, EDIT
                .EXTN   FORM.,BACK.     ; TICKET FORMAT BLOCKS
                .EXTN   GET.,PUT.       ; FORMATTED INPUT/OUTPUT
                .EXTN   CVA.,CVB.       ; CONVERSION SUBROUTINES
                .EXTN   MPY.,DIV.       ; ARITHMETIC SUBROUTINES
                .EXTN   US0.,TS0.       ; STATUS TABLE DISPLAY
                .EXTN   US1.,TS1.       ; STATUS TABLE INSERTION
                .EXTN   US2.,TS2.       ; STATUS TABLE DELETION
                .EXTN   US3.,TS3.       ; STATUS TABLE LOCATION
                .EXTN   TYPE.,ASK.      ; DATA STRING INPUT, OUTPUT
                .EXTN   CCN.            ; CONTROL CENTER ACCESS
                .EXTN   LOG.            ; TRANSACTION LOG
                .EXTN   KILL.           ; KILL NON-RESIDENT
                .EXTN   OLAY.           ; LOAD A UTILITY
                .EXTN   FALT.           ; FIND A TICKET
                .EXTN   PALT.           ; FIND NEW PAGE
                .EXTN   CALT.           ; PRINT A FILE
                .EXTN   DATA.           ; DATA FIELD OUTPUT
                .EXTN   TERM.           ; DATA FIELD TERMINATE
                .EXTN   TAB.            ; DATA FIELD ADVANCE
                .EXTN   TOP.            ; INSTALL FORMAT EDIT
                .EXTN   S.TOP,S.END     ; STATUS TABLE TERMINI
                .EXTN   C.TOP,C.END     ; CONTROL TABLE TERMINI

                .ENT    .US0,.TS0       ; STATUS TABLE OUTPUT VECTORS
                .ENT    .US1,.TS1       ; STATUS TABLE INSERT VECTORS
                .ENT    .US2,.TS2       ; STATUS TABLE DELETE VECTORS
                .ENT    .US3,.TS3       ; STATUS TABLE LOCATE VECTORS
                .ENT    .MPY,.DIV       ; ARITHMETIC FUNCTION VECTORS
                .ENT    .CVA,.CVB       ; CONVERSION FUNCTION VECTORS
                .ENT    .GET,.PUT       ; FORMATTED DATA INPUT/OUTPUT
                .ENT    .DUMP,.EDIT     ; FORMAT DISPLAY, EDIT
                .ENT    .FORM,.BACK     ; TICKET FORMAT BLOCKS
                .ENT    TYPE,ASK        ; DATA STRING INPUT, OUTPUT
                .ENT    .LOG            ; TRANSACTION LOG
                .ENT    .CCN            ; CONTROL CENTER ACCESS
                .ENT    .KILL           ; KILL NON-RESIDENT CODE
                .ENT    .CALL           ; LOAD NON-RESIDENT CODE
                .ENT    .FIND           ; FIND TICKET BY NUMBER
                .ENT    .PAGE           ; FIND NEXT TICKET PAGE
                .ENT    .DATA           ; ENTER AND DISPLAY DATA
                .ENT    .TERM           ; TERMINATE DATA FIELD
                .ENT    .TAB            ; ADVANCE TO NEXT FIELD
                .ENT    .TOP            ; INSTALL FORMAT EDIT
                .ENT    NEXT            ; NEXT AVAILABLE TICKET
                .ENT    LIST            ; DATA BLOCK LIST POINTER
                .ENT    ZONE            ; CURRENT TIME ZONE (ASCII)
                .ENT    USTB,USTE       ; UNIT STATUS TABLE TERMINI
                .ENT    TSTB,TSTE       ; TICKET STATUS TABLE TERMINI
                .ENT    CCTB,CCTE       ; CONTROL CENTER TABLE TERMINI
                .ENT    TAD,SAD,MAD,FAD ; SCREEN ADDRESSES
                .ENT    LRF,LTF         ; LEADS TRANSMIT/RECEIVE FLAGS
                .ENT    SSB             ; SAVE STATUS BLOCK
```

```
                .ZREL   ; AIDS SYSTEM VECTOR TABLE

    000040-177777 .MPY:   MPY.    ; MULTIPLY FUNCTION
    000041-177777 .DIV:   DIV.    ; DIVIDE FUNCTION
    000042-177777 .CVA:   CVA.    ; ASCII CONVERSION
    000043-177777 .CVB:   CVB.    ; BINARY CONVERSION
    000044-177777 TYPE:   TYPE.   ; TEXT STRING TYPER
    000045-177777 ASK:    ASK.    ; DATA STRING INPUT
    000046-177777 .GET:   GET.    ; INPUT IN FORMAT
    000047-177777 .PUT:   PUT.    ; OUTPUT IN FORMAT

    000050-177777 .DUMP:  DUMP.   ; DISPLAY IN FORMAT
    000051-177777 .EDIT:  EDIT.   ; FORMAT EDIT TABLE
    000052-177777 .FORM:  FORM.   ; TICKET FORMAT BLOCK
    000053-177777 .BACK:  BACK.   ; REVERSE SIDE FORMAT
    000054-177777 .DATA:  DATA.   ; DATA FIELD DISPLAY
    000055-177777 .TERM:  TERM.   ; DATA FIELD TERMINATE
    000056-177777 .TAB:   TAB.    ; DATA FIELD ADVANCE
    000057-177777 .TOP:   TOP.    ; START FORMAT EDIT

    000060-177777 .US0:   US0.    ; UNIT STATUS TABLE OUTPUT
    000061-177777 .TS0:   TS0.    ; TICKET STATUS TABLE OUTPUT
    000062-177777 .US1:   US1.    ; UNIT STATUS TABLE INSERT
    000063-177777 .TS1:   TS1.    ; TICKET STATUS TABLE INSERT
    000064-177777 .US2:   US2.    ; UNIT STATUS TABLE DELETE
    000065-177777 .TS2:   TS2.    ; TICKET STATUS TABLE DELETE
    000066-177777 .US3:   US3.    ; UNIT STATUS TABLE LOCATE
    000067-177777 .TS3:   TS3.    ; TICKET STATUS TABLE LOCATE

    000070-177777 .CCN:   CCN.    ; LOCATE CENTER NAME
    000071-000000         0       ; EXTRA WORD
    000072-177777 .FIND:  FALT.   ; FIND TICKET BY NUMBER
    000073-177777 .PAGE:  PALT.   ; FIND NEXT TICKET PAGE
    000074-177777 .CALL:  OLAY.   ; LOAD UTILITY PROGRAM
    000075-177777 .KILL:  KILL.   ; DELETE NON-RESIDENT CODE
    000076-177777 .COPY:  CALT.   ; COPY FILE TO PRINTER
    000077-177777 .LOG:   LOG.    ; TRANSACTION LOG ENTRY

    000100-000400 TAD:    11.*400 ; SCREEN ADDRESS FOR TAGS
    000101-000400 SAD:    13.*400 ; SCREEN ADDRESS FOR STATUS
    000102-013000 MAD:    22.*400 ; SCREEN ADDRESS FOR MESSAGES
    000103-013400 FAD:    23.*400 ; SCREEN ADDRESS FOR FLAGS

    000104-000000 LIST:   0       ; BLOCK LIST POINTER
    000105-041504 ZONE:   "C*400+"D ; TIME ZONE (ASCII)

    000106-000000- SSB:   CCTB
    000107-000000-        NEXT
    000000-177777 CCTB:   C.TOP   ; CONTROL CENTER TABLE BEGIN
    000001-177777 CCTE:   C.END   ; CONTROL CENTER TABLE END
    000002-177777 USTB:   S.TOP   ; UNIT STATUS TABLE BEGIN
    000003-000062- USTE:  S.TOP   ; UNIT STATUS TABLE END
    000004-177777 TSTB:   S.END   ; TICKET STATUS TABLE BEGIN
    000005-000064- TSTE:  S.END   ; TICKET STATUS TABLE END
    000006-000000         00
    000007-000000 NEXT:   0       ; NEXT AVAILABLE TICKET
```



```
1   000060-000000 LRF:    0       ; LEADS RECEIVE FILE FLAGS
2   000061-000000 LTF:    0       ; LEADS TRANSMIT FILE FLAGS
3
4                 .END
```

SYMBOL	VALUE	DEF'N	REFERENCES	
[illegible]	000005-	2:08	1:38	
[illegible].	-X	1:14	2:08	
[illegible].	-X	1:06	2:15	
[illegible].	-X	1:21	2:36	
[illegible].	-X	1:15	2:30	
[illegible]	000050-	2:49	1:54	2:47
[illegible]	000051-	2:50	1:54	
[illegible].	-X	1:08	2:05	
[illegible].	-X	1:08	2:06	
[illegible]	-X	1:27	2:50	
[illegible]	-X	1:27	2:49	
[illegible].	-X	1:22	2:16	
[illegible].	-X	1:09	2:04	
[illegible].	-X	1:05	2:12	
[illegible].	-X	1:05	2:13	
[illegible]	000043-	2:42	1:55	
[illegible].	-X	1:19	2:32	
[illegible].	-X	1:06	2:14	
[illegible].	-X	1:07	2:09	
[illegible].	-X	1:17	2:35	
[illegible]	000044-	2:44	1:50	
[illegible].	-X	1:16	2:37	
[illegible]	000060-	3:41	1:56	
[illegible]	000061-	3:42	1:56	
[illegible]	000042-	2:41	1:55	
[illegible]	-X	1:49	2:03	
[illegible]	000057-	2:56	1:49	2:48
[illegible].	-X	1:18	2:34	
[illegible].	-X	1:20	2:33	
[illegible].	-X	1:07	2:10	
[illegible]	000041-	2:40	1:55	
[illegible]	000046-	2:47	1:57	
[illegible]	-X	1:26	2:53	2:54
[illegible]	-X	1:26	2:51	2:52
[illegible]	-X	1:24	2:18	
[illegible]	000040-	2:39	1:55	
[illegible].	-X	1:23	2:17	
[illegible].	-X	1:25	2:19	
[illegible].	-X	1:10	2:22	
[illegible].	-X	1:11	2:24	
[illegible].	-X	1:12	2:26	
[illegible].	-X	1:13	2:28	
[illegible]	000054-	2:53	1:53	
[illegible]	000055-	2:54	1:53	
[illegible]	000004-	2:07	1:38	
[illegible].	-X	1:14	2:07	
[illegible].	-X	1:10	2:21	
[illegible].	-X	1:11	2:23	
[illegible].	-X	1:12	2:25	
[illegible].	-X	1:13	2:27	
[illegible]	000052-	2:51	1:52	
[illegible]	000053-	2:52	1:52	
[illegible]	000045-	2:45	1:51	
[illegible]	000013-	2:15	1:37	
[illegible]	000034-	2:31	1:42	

SYMBOL	VALUE	DEF'N	REFERENCES
CCN	000030-	2:30	1:40
COPY	000036-	2:36	
CVA	000002-	2:05	1:34
CVB	000003-	2:06	1:34
DATA	000014-	2:16	1:45
DIV	000001-	2:04	1:33
DUMP	000010-	2:12	1:36
EDIT	000011-	2:13	1:36
FIND	000032-	2:32	1:43
FORM	000012-	2:14	1:37
GET	000006-	2:09	1:35
KILL	000035-	2:35	1:41
LOG	000037-	2:37	1:39
MPY	000000-	2:03	1:33
PAGE	000033-	2:33	1:44
PUT	000007-	2:10	1:35
TAB	000016-	2:18	1:47
TERM	000015-	2:17	1:46
TOP	000017-	2:19	1:48
TS0	000021-	2:22	1:29
TS1	000023-	2:24	1:30
TS2	000025-	2:26	1:31
TS3	000027-	2:28	1:32
US1	000020-	2:21	1:29
US1	000022-	2:23	1:30
US	000024-	2:25	1:31
US3	000026-	2:27	1:32

FLAGGED LINES: NONE

CONTROL CENTER TABLE

The control center table contains parameters relevant to each control center. Each control center has its own 40 word (octal) entry in the control center table as follows:

WORD	USE
0	Control center name
1	Department name
2	File sequence identifier
3	File sequence number (last used)
4	LEADS call direction code (CDC)
5	LEADS drop code
6	Radio frequency channel number
7	Printer deferral code
10	Printer file flag
11	Printer TIT pointer
12 - 17	Empty but reserved
20 - 27	16 character city name
30	State (2 ASCII characters)
31 - 37	Unassigned (for use by report generators)

The location and size of the control center table is defined in the system vector table, SVT.

```
PAGE  1          A I D S     S Y S T E M     S T A T U S     T A B L E
02/20/74
  1                    .HEAD   A I D S     S Y S T E M     S T A T U S     T A B
  2
  3                    .TITLE  SST
  4
  5                    .EXTN   TIT20,TIT21,TIT22,TIT23
  6
  7                    .ENT    C.TOP,C.END
  8                    .ENT    S.TOP,S.END
  9          000001    .TXTM   1
 10          000001    .TXTN   1
 11          000001    .TXTX   1
 12                    .NREL
 13
 14    000000'         C.TOP:
 15
 16    000000'047400   CC1:    "0*400              ; CONTROL CENTER NAME
 17    000001'047520           "0*400+"P           ; DEPARTMENT NAME
 18    000002'033464           "7*400+"4           ; FILE CONTROL SEQUENCE
 19    000003'000003           0                   ; FILE CONTROL NUMBER
 20    000004'042111           "D*400+"I           ; LEADS "CDC" NAME
 21    000005'000110           "H                  ; LEADS DROP-CODE
 22    000006'000001           1                   ; RADIO CHANNEL NUMBER
 23    000007'000000           0                   ; PRINTER DEFERRAL
 24
 25    000010'000000           0                   ; PRINTER FILE FLAGS
 26    000011'177777           TIT20               ; PRINTER TIT POINTER
 27          000006  .REP      CC1+20-.
 28    000012'000000           0
 29    000020'047501   .TXT    "OAK PARK           IL"
 30          000007  .REP      CC1+40-.
 31    000031'000000           0
 32
 33    000040'043000   CC2:    "F*400              ; CONTROL CENTER NAME
 34    000041'043120           "F*400+"P           ; DEPARTMENT NAME
 35    000042'033464           "7*400+"4           ; FILE CONTROL SEQUENCE
 36    000043'000003           0                   ; FILE CONTROL NUMBER
 37    000044'042107           "D*400+"G           ; LEADS "CDC" NAME
 38    000045'000103           "C                  ; LEADS DROP-CODE
 39    000046'000001           1                   ; RADIO CHANNEL NUMBER
 40    000047'000003           0                   ; PRINTER DEFERRAL
 41
 42    000050'000000           0                   ; PRINTER FILE FLAGS
 43    000051'177777           TIT22               ; PRINTER TIT POINTER
 44          000006  .REP      CC2+20-.
 45    000052'000000           0
 46    000060'043117   .TXT    "FOREST PARK        IL"
 47          000007  .REP      CC2+40-.
 48    000071'000000           0
```

```
000000' 051000    CC3:    "R*400          ; CONTROL CENTER NAME
000001' 051106            "R*400+"F       ; DEPARTMENT NAME
000002' 033464            "7*400+"4       ; FILE CONTROL SEQUENCE
000003' 000000            0               ; FILE CONTROL NUMBER
000004' 042101            "D*400+"A       ; LEADS "CDC" NAME
000005' 000107            "G              ; LEADS DROP-CODE
000006' 000001            1               ; RADIO CHANNEL NUMBER
000007' 000000            0               ; PRINTER DEFERRAL

000010' 000000            0               ; PRINTER FILE FLAGS
000011' 177777            TIT23           ; PRINTER TIT POINTER
000006          .REP      CC3+20-.
000012' 000000            0
000013' 051111    .TXT    "RIVER FOREST    IL"
000007          .REP      CC3+40-.
000031' 000003            0

000140'         C.END:

                ; STATUS TABLE ALLOCATION

000004          DEPTS=  4.        ; NUMBER OF DEPARTMENTS
000020          UNITS=  16.       ; AVERAGE NUMBER OF UNITS
000020          FORMS=  UNITS     ; NORMAL NUMBER OF TICKETS
000040          ENTRY=  40        ; STATUS TABLE ENTRY SIZE

010000          SIZE=   UNITS+FORMS*DEPTS*ENTRY ; TOTAL TABLE SIZE

                .NREL

000140'         S.TOP:

010000'         .LOC    C.TOP+SIZE      ; UNIT/TICKET STATUS TABLE

100000'         S.END:

                .END
```


SYMBOL	VALUE	DEF'N	REFERENCES	
CC1	000000'	1:16	1:27	1:30
CC2	000040'	1:33	1:44	1:47
CC3	000100'	2:01	2:12	2:15
C.END	000140'	2:18	1:07	
C.TOP	000000'	1:14	1:07	2:33
DEPTS	000004	2:22	2:27	
ENTRY	000040	2:25	2:27	
FORMS	000020	2:24	2:27	
SIZE	010000	2:27	2:33	
S.END	010000'	2:35	1:08	
S.TOP	000140'	2:31	1:08	
TIT20	'X	1:05	1:26	
TIT21	X	1:05		
TIT22	'X	1:05	1:43	
TIT23	'X	1:05	2:11	
UNITS	000020	2:23	2:24	2:27

FLAGGED LINES: NONE

TERMINAL INFORMATION TABLES

Associated with each terminal is a terminal information table (TIT), which is 14ø octal words long. The first 4ø words are fairly strictly defined in their usage. The remaining 1øø words are a bit less rigorously defined, but to allow communication between routines they are more or less defined by convention.

The first ten octal words are permanently defined for each terminal and contain information as follows:

WORD NUMBER	NAME	DESCRIPTION
ø	ID	A unique octal identifier for each terminal
1	FLAG	Defines the type of terminal (i.e. printer or display screen)
2	LINK	Pointer to TIT of next terminal
3	LOOP	Extra link word -- normally points to the AWAIT routine
4	ZIP	Keyboard input vector
5	ZEP	Device input handler
6	ZAP	Display output vector
7	ZOP	Device output handler

Words 1ø-27 are dynamic vectors and parameters.

10	EDIT	Format entry table base
11	SPOT	Current cursor location
12	CHAR	Current input character
13	HOME	Official cursor position
14	FORM	Format block address
15	DATA	Data block address
16	FIELD	Format field number
17	INDEX	Data character index (within field)

(continued)

WORD NUMBER	NAME	DESCRIPTION
2ø	CCN	Control center name
21	OPR	Operator identifier
22	TIME	Current time of day
23	UNIT	Current unit number
24	TKT	Current ticket number
25	OLD	Previous ticket number
26	RUN	Non-resident code vector
27	END	End control word (vector)

Words 3ø-33 are output parameters:

3ø	MODE	Terminal display mode
31	PADS	Padding control count
32	LINE	Line length in spaces
33	PAGE	Page length in lines

Words 34-37 are return address storage locations for input-output routines:

34	RZIP	Keyboard input routine return
35	RZEP	Device input handler return
36	RZAP	Display output routine return
37	RZOP	Device output handler return

Words 4ø-77 are rather variable but break down into four main groups:

4ø-47	Pø-P7	Parameter words used for inter-routine communication
5ø-57	Rø-R7	Return address storage words
6ø-67	Sø-S7	Routine storage words
7ø-77	Tø-T7	Temporary storage words -- not to be trusted around external calls

(continued)

The remaining 4ø words are used for text string storage.

The address of the terminal information table is almost universally kept in AC2 and if necessary, AC2 can be used and reset to the table address from word 2 of the system.


```
                .HEAD   A I D S      -       TERMINAL INFORMATION TABLES

000000  ID=     0       ; TERMINAL IDENTIFIER
000001  FLAG=   1       ; TERMINAL STATUS FLAGS
000002  LINK=   2       ; TIT CHAIN LINK WORD
000003  LOOP=   3       ; EXTRA LINK WORD
000004  ZIP=    4       ; KEYBOARD INPUT VECTOR
000005  ZEP=    5       ; DEVICE INPUT HANDLER
000006  ZAP=    6       ; DISPLAY OUTPUT VECTOR
000007  ZOP=    7       ; DEVICE OUTPUT HANDLER

000010  EDIT=   10      ; FORMAT ENTRY TABLE BASE
000011  SPOT=   11      ; CURRENT CURSOR LOCATION
000012  CHAR=   12      ; CURRENT INPUT CHARACTER
000013  HOME=   13      ; OFFICIAL CURSOR POSITION
000014  FORM=   14      ; FORMAT BLOCK ADDRESS
000015  DATA=   15      ; DATA BLOCK ADDRESS
000016  FIELD=  16      ; FORMAT FIELD NUMBER
000017  INDEX=  17      ; DATA CHARACTER INDEX

000020  CCN=    20      ; CONTROL CENTER NAME
000021  OPR=    21      ; OPERATOR IDENTIFIER
000022  TIME=   22      ; CURRENT TIME OF DAY
000023  UNIT=   23      ; CURRENT UNIT NUMBER
000024  TKT=    24      ; CURRENT TICKET NUMBER
000025  OLD=    25      ; PREVIOUS TICKET NUMBER
000026  RUN=    26      ; NON-RESIDENT CODE
000027  END=    27      ; END CONTROL WORD

000034  MODE=   34      ; TERMINAL DISPLAY MODE
000035  PADS=   35      ; PADDING CONTROL COUNT
000036  LINE=   36      ; LINE LENGTH IN SPACES
000037  PAGE=   37      ; PAGE LENGTH IN LINES

                .ENT    ID,FLAG
                .ENT    LINK,LOOP
                .ENT    EDIT,CHAR
                .ENT    SPOT,HOME
                .ENT    FORM,DATA
                .ENT    FIELD,INDEX
                .ENT    MODE,PADS
                .ENT    LINE,PAGE
                .ENT    ZIP,ZEP
                .ENT    ZAP,ZOP
                .ENT    CCN,OPR
                .ENT    TKT,OLD
                .ENT    UNIT
                .ENT    TIME
                .ENT    RUN
                .ENT    END
```

```
                        .TITLE  TIT

                        .EXTN   .TTI,.TTO
                        .EXTN   .MI1,.MO1
                        .EXTN   .MI2,.MO2
                        .EXTN   .MI3,.MO3
                        .EXTN   TZEP,TZOP
                        .EXTN   DZIP,DZAP
                        .EXTN   CZIP,PZAP
                        .EXTN   LZIP,LZAP
                        .EXTN   AWAIT
                        .EXTN   PRINT
                        .EXTN   LEADS
                        .EXTN   WATCH
                        .EXTN   SLOB

                        .EXTD   CCTB,CCTE

        000140  SIZE=   140             ; TOTAL BLOCK LENGTH

        000006  PAC2=   6               ; PROCESS AC2 STORAGE
        000012  PLOC=   12              ; PROCESS LOCAL BLOCK

        100000  DFLAG=  @0              ; DISPLAY UNIT FLAGS
        000000  PFLAG=  0               ; PRINTER UNIT FLAGS
        000000  LFLAG=  0               ; LINE CONTROL FLAGS

                        .ENT    CTIT,LTIT
                        .ENT    TITI0,TITI1,TITI2,TITI3
                        .ENT    TITI4,TITI5,TITI6,TITI7
                        .ENT    TIT20,TIT21,TIT22,TIT23
                        .NREL

                        ; TERMINAL BLOCKS

  000000'000004 CTIT:   4               ; CONSOLE TERMINAL
  000001'000000         PFLAG           ; IT IS A PRINTER
  000002'000140'        TITI0           ; LINK TO NEXT TIT
  000003'177777         AWAIT           ; PROCESS START
  000004'177777         CZIP            ; CONSOLE INPUT
  000005'177777         TZEP            ; INPUT DRIVER
  000006'177777         PZAP            ; PRINTER OUTPUT
  000007'177777         TZOP            ; OUTPUT DRIVER

        000025  .REP    CTIT+LINE-.     ; CONTROL WORDS
  000012'000000         0
  000036'000170         120.            ; LINE LENGTH
  000037'000074         60.             ; PAGE LENGTH
```



```
  1  00040'000000  SAVE:   0               ; TABLE ADDRESS SAVE WORD
  2  00041'000040  SPIT:   40              ; CONTROL TABLE ENTRY SIZE
  3  00042'000000' TITS:   CTIT            ; CONSOLE TERMINAL TABLE
  4  00043'002454' STAT:   FILE            ; STATUS TABLE FILE NAME
  5  00044'002462' FLOG:   NAME            ; SYSTEM LOG FILE NAME
  6  00045'177777  BUFF:   SLOB            ; SYSTEM LOG OUTPUT BUFFER
  7  00046'177777  DUMP:   PRINT           ; PRINTER MANAGER START ADDRESS
  8
  9                     ; INITIALIZE STATUS AND LOG
 10
 11  00047'030774  START:  LDA 2   STAT    ; GET FILE NAME
 12  00050'007400          JSR     @D.CYS,3; CALL OVERLAY
 13  00051'000014          0.LOD           ; LOAD STATUS TABLES
 14
 15  00052'030772          LDA 2   FLOG    ; GET FILE NAME
 16  00053'024772          LDA 1   BUFF    ; BUFFER ADDRESS
 17  00054'102000          ADC     0,0     ; OPEN OPTION
 18  00055'007400          JSR     @D.CYS,3; CALL OVERLAY
 19  00056'000006          0.OPN           ; OPEN LOG AT END
 20
 21                     ; INITIATE PRINTER MANAGERS
 22
 23  00057'030001$         LDA 2   CCTB    ; CONTROL TABLE TOP
 24  00060'050760  MORE:   STA 2   SAVE    ; SAVE FOR LATER ON
 25  00061'034765          LDA 3   DUMP    ; GET START ADDRESS
 26  00062'177240          ADDOR   3,3     ; SET INDIRECT BIT
 27  00063'054447          STA 3   .PCB    ; ON START ADDRESS
 28  00064'030443          LDA 2   .IOC    ; I/O CONTROL BLOCK
 29  00065'006044          .IO             ; START A PROCESS
 30  00066'006047          ABORT           ; MUST NEVER OCCUR
 31
 32  00067'034443          LDA 3   .PCB    ; NEW PROCESS BLOCK
 33  00070'030750          LDA 2   SAVE    ; CONTROL CENTER BLOCK
 34  00071'051412          STA 2   PLOC,3  ; LOCAL BLOCK ADDRESS
 35  00072'051406          STA 2   PAC2,3  ; ALSO PUT IN AC2
 36
 37  00073'020746          LDA 0   SPIT    ; TABLE ENTRY LENGTH
 38  00074'113000          ADD     0,2     ; MOVE TO NEXT ENTRY
 39  00075'020002$         LDA 0   CCTE    ; CONTROL TABLE END
 40  00076'112512          SLE     0,2     ; IF MORE CENTERS,
 41  00077'000761          JMP     MORE    ; GO BACK AGAIN
```

```
                        ; INITIATE INPUT MONITORS

00130'030742            LDA 2   TITS    ; INITIAL TERMINAL
00131'050737  NEXT:     STA     2,SAVE  ; SAVE TERMINAL BLOCK
00132'035003            LDA     3,LOOP,2; PROCESS START
00133'175122            MOVZL   3,3 SZC ; IF NOT ADDRESS,
00134'000415            JMP     TEST    ; START NO PROCESS
00135'175240            MOVOR   3,3     ; SET INDIRECT BIT
00136'054424            STA     3,.PCB  ; ON START ADDRESS

00137'126400            SUB     1,1
00140'032425            LDA 0   .OFF    ; CMEAR TERMINAL
00141'047006            JSR     @ZAP,2

00142'030415            LDA     2,.IOC  ; I/O CONTROL BLOCK
00143'006044            .IO             ; START A PROCESS
00144'006047            ABORT           ; MUST NEVER OCCUR

00145'034415            LDA     3,.PCB  ; NEW PROCESS BLOCK
00146'030722            LDA     2,SAVE  ; CURRENT TIT BLOCK
00147'051412            STA     2,PLOC,3; LOCAL BLOCK ADDRESS
00150'051406            STA     2,PAC2,3; ALSO PUT IN AC2

00151'031002  TEST:     LDA     2,LINK,2; NEXT TIT
00152'020720            LDA     0,TITS  ; INITIAL ONE
00153'112414            SEQ     0,2     ; IF NOT DONE,
00154'000755            JMP     NEXT    ; REPEAT

00155'002401            JMP     @.+1    ; JUMP TO PROCESS
00156'177777            WATCH           ; UPDATE MONITOR

00157'000133'  .IOC:    .+1             ; I/O CONTROL BLOCK
00160'000000            0
00161'000004            4
00162'000003  .PCB:     0
00163'000000            0
00164'000000            0

00165'140000  .OFF:     140000          ; CLEAR CONTROL WORD

000002        .BLK      CTIT+SIZE-.     ; REST OF WORKING AREA
```



```
00140'000010  TITI0:    10              ; TERMINAL 10
00141'100000            DFLAG           ; IT IS A DISPLAY
00142'000300'           TITI1           ; LINK TO NEXT TIT
00143'000003'           AWAIT           ; PROCESS START
00144'177777            DZIP            ; DISPLAY INPUT
00145'000005'           TZEP            ; INPUT DRIVER
00146'177777            DZAP            ; DISPLAY OUTPUT
00147'000007'           TZOP            ; OUTPUT DRIVER

000010        .REP      10              ; CONTROL WORDS
00150'000000            0
00160'047520            "O*400+"P       ; CENTER NAME
000015        .REP      TITI0+LINE-.    ; CONTROL WORDS
00161'000000            0
00176'000120            80.             ; LINE LENGTH
00177'000030            24.             ; PAGE LENGTH
000100        .BLK      TITI0+SIZE-.    ; WORKING AREA

00300'000011  TITI1:    11              ; TERMINAL 11
00301'100000            DFLAG           ; IT IS A DISPLAY
00302'000440'           TITI2           ; LINK TO NEXT TIT
00303'000143'           AWAIT           ; PROCESS START
00304'000144'           DZIP            ; DISPLAY INPUT
00305'000145'           TZEP            ; INPUT DRIVER
00306'000146'           DZAP            ; DISPLAY OUTPUT
00307'000147'           TZOP            ; OUTPUT DRIVER

000010        .REP      10              ; CONTROL WORDS
00310'000000            0
00320'047520            "O*400+"P       ; CENTER NAME
000015        .REP      TITI1+LINE-.    ; CONTROL WORDS
00321'000000            0
00336'000120            80.             ; LINE LENGTH
00337'000030            24.             ; PAGE LENGTH
000100        .BLK      TITI1+SIZE-.    ; WORKING AREA
```

```
                        TIT12:  12              ; TERMINAL 12
                                DFLAG           ; IT IS A DISPLAY
                                TIT13           ; LINK TO NEXT TIT
                                AWAIT           ; PROCESS START
                                DZIP            ; DISPLAY INPUT
                                TZEP            ; INPUT DRIVER
                                DZAP            ; DISPLAY OUTPUT
                                TZOP            ; OUTPUT DRIVER

                        .REP    10              ; CONTROL WORDS
                                0
                                "O*400+"P       ; CENTER NAME
                        .REP    TIT12+LINE-.    ; CONTROL WORDS
                                0
                                80.             ; LINE LENGTH
                                24.             ; PAGE LENGTH
                        .BLK    TIT12+SIZE-.    ; WORKING AREA

                        TIT13:  13              ; TERMINAL 13
                                DFLAG           ; IT IS A DISPLAY
                                TIT14           ; LINK TO NEXT TIT
                                AWAIT           ; PROCESS START
                                DZIP            ; DISPLAY INPUT
                                TZEP            ; INPUT DRIVER
                                DZAP            ; DISPLAY OUTPUT
                                TZOP            ; OUTPUT DRIVER

                        .REP    10              ; CONTROL WORDS
                                0
                                "O*400+"P       ; CENTER NAME
                        .REP    TIT13+LINE-.    ; CONTROL WORDS
                                0
                                80.             ; LINE LENGTH
                                24.             ; PAGE LENGTH
                        .BLK    TIT13+SIZE-.    ; WORKING AREA
```



```
00740'000014    TIT14:  14              ; TERMINAL 14
00741'100000            DFLAG           ; IT IS A DISPLAY
00742'001100'           TIT15           ; LINK TO NEXT TIT
00743'000603'           AWAIT           ; PROCESS START
00744'000604'           DZIP            ; DISPLAY INPUT
00745'000605'           TZEP            ; INPUT DRIVER
00746'000606'           DZAP            ; DISPLAY OUTPUT
00747'000607'           TZOP            ; OUTPUT DRIVER

        000010  .REP    10              ; CONTROL WORDS
00750'000000            0
00760'043120            "F*400+"P       ; CENTER NAME
        000015  .REP    TIT14+LINE-.    ; CONTROL WORDS
00761'000000            0
00776'000120            80.             ; LINE LENGTH
00777'000030            24.             ; PAGE LENGTH
        000100  .BLK    TIT14+SIZE-.    ; WORKING AREA

01100'000015    TIT15:  15              ; TERMINAL 15
01101'100000            DFLAG           ; IT IS A DISPLAY
01102'001240'           TIT16           ; LINK TO NEXT TIT
01103'000743'           AWAIT           ; PROCESS START
01104'000744'           DZIP            ; DISPLAY INPUT
01105'000745'           TZEP            ; INPUT DRIVER
01106'000746'           DZAP            ; DISPLAY OUTPUT
01107'000747'           TZOP            ; OUTPUT DRIVER

        000010  .REP    10              ; CONTROL WORDS
01110'000000            0
01120'043120            "F*400+"P       ; CENTER NAME
        000015  .REP    TIT15+LINE-.    ; CONTROL WORDS
01121'000000            0
01136'000120            80.             ; LINE LENGTH
01137'000030            24.             ; PAGE LENGTH
        000100  .BLK    TIT15+SIZE-.    ; WORKING AREA
```

```
01240'000016  TIT16:  16              ; TERMINAL 16
01241'100000          DFLAG           ; IT IS A DISPLAY
01242'001400'         TIT17           ; LINK TO NEXT TIT
01243'001103'         AWAIT           ; PROCESS START
01244'001104'         DZIP            ; DISPLAY INPUT
01245'001105'         TZEP            ; INPUT DRIVER
01246'001106'         DZAP            ; DISPLAY OUTPUT
01247'001107'         TZOP            ; OUTPUT DRIVER

      000010  .REP    10              ; CONTROL WORDS
01250'000000          0
01260'051106          "R*400+"F       ; CENTER NAME
      000015  .REP    TIT16+LINE-.    ; CONTROL WORDS
01261'000000          0
01276'000120          80.             ; LINE LENGTH
01277'000030          24.             ; PAGE LENGTH
      000100  .BLK    TIT16+SIZE-.    ; WORKING AREA

01400'000017  TIT17:  17              ; TERMINAL 17
01401'100000          DFLAG           ; IT IS A DISPLAY
01402'001540'         TIT20           ; LINK TO NEXT TIT
01403'001243'         AWAIT           ; PROCESS START
01404'001244'         DZIP            ; DISPLAY INPUT
01405'001245'         TZEP            ; INPUT DRIVER
01406'001246'         DZAP            ; DISPLAY OUTPUT
01407'001247'         TZOP            ; OUTPUT DRIVER

      000010  .REP    10              ; CONTROL WORDS
01410'000000          0
01420'051106          "R*400+"F       ; CENTER NAME
      000015  .REP    TIT17+LINE-.    ; CONTROL WORDS
01421'000000          0
01436'000120          80.             ; LINE LENGTH
01437'000030          24.             ; PAGE LENGTH
      000100  .BLK    TIT17+SIZE-.    ; WORKING AREA
```



```
01540'000020  TIT20:  20              ; TERMINAL 20
01541'000000          PFLAG           ; IT IS A PRINTER
01542'001700'         TIT21           ; LINK TO NEXT TIT
01543'100000          @0              ; NO PROCESS
01544'000004'         CZIP            ; CONSOLE INPUT
01545'001405'         TZEP            ; INPUT DRIVER
01546'000006'         PZAP            ; PRINTER OUTPUT
01547'001407'         TZOP            ; OUTPUT DRIVER

      000010  .REP    10              ; CONTROL WORDS
01550'000000          0
01560'047400          "O*400          ; CENTER NAME
      000015  .REP    TIT20+LINE-.    ; CONTROL WORDS
01561'000000          0
01576'000120          80.             ; LINE LENGTH
01577'000074          60.             ; PAGE LENGTH
      000100  .BLK    TIT20+SIZE-.    ; WORKING AREA

01700'000021  TIT21:  21              ; TERMINAL 21
01701'000000          PFLAG           ; IT IS A PRINTER
01702'002040'         TIT22           ; LINK TO NEXT TIT
01703'100000          @0              ; NO PROCESS
01704'001544'         CZIP            ; CONSOLE INPUT
01705'001545'         TZEP            ; INPUT DRIVER
01706'001546'         PZAP            ; PRINTER OUTPUT
01707'001547'         TZOP            ; OUTPUT DRIVER

      000010  .REP    10              ; CONTROL WORDS
01710'000000          0
01720'047400          "O*400          ; CENTER NAME
      000015  .REP    TIT21+LINE-.    ; CONTROL WORDS
01721'000000          0
01736'000120          80.             ; LINE LENGTH
01737'000074          60.             ; PAGE LENGTH
      000100  .BLK    TIT21+SIZE-.    ; WORKING AREA
```

```
02040'000022  TIT22:  22              ; TERMINAL 22
02041'000000          PFLAG           ; IT IS A PRINTER
02042'002200'         TIT23           ; LINK TO NEXT TIT
02043'100000          @0              ; NO PROCESS
02044'001704'         CZIP            ; CONSOLE INPUT
02045'001705'         TZEP            ; INPUT DRIVER
02046'001706'         PZAP            ; PRINTER OUTPUT
02047'001707'         TZOP            ; OUTPUT DRIVER

      000010  .REP     10            ; CONTROL WORDS
02050'000000          0
02060'043000          "F*400          ; CENTER NAME
      000015  .REP     TIT22+LINE-.  ; CONTROL WORDS
02061'000000          0
02076'000120          80.             ; LINE LENGTH
02077'000074          60.             ; PAGE LENGTH
      000100  .BLK     TIT22+SIZE-.  ; WORKING AREA

02230'000023  TIT23:  23              ; TERMINAL 23
02231'000000          PFLAG           ; IT IS A PRINTER
02232'002340'         TIT24           ; LINK TO NEXT TIT
02233'100000          @0              ; NO PROCESS
02234'002044'         CZIP            ; CONSOLE INPUT
02235'002045'         TZEP            ; INPUT DRIVER
02236'002046'         PZAP            ; PRINTER OUTPUT
02237'002047'         TZOP            ; OUTPUT DRIVER

      000010  .REP     10            ; CONTROL WORDS
02240'000000          0
02250'051000          "R*400          ; CENTER NAME
      000015  .REP     TIT23+LINE-.  ; CONTROL WORDS
02241'000000          0
02256'000120          80.             ; LINE LENGTH
02257'000074          60.             ; PAGE LENGTH
      000100  .BLK     TIT23+SIZE-.  ; WORKING AREA
```



```
1   02340'          LTIT:
2   02340'000024    TIT24:  24            ; TERMINAL 24
3   02341'000000            LFLAG         ; LEADS LINE CONTROL
4   02342'000000'           CTIT          ; LINK TO FIRST TIT
5   02343'177777            LEADS         ; PROCESS START
6   02344'177777            LZIP          ; LEADS INPUT
7   02345'002205'           TZEP          ; INPUT DRIVER
8   02346'177777            LZAP          ; LEADS OUTPUT
9   02347'002201'           TZOP          ; OUTPUT DRIVER
10
11        000030  .REP      LTIT+40-.     ; CONTROL WORDS
12  02350'000000            0
13
14                  ; SETUP INTERRUPT VECTORS
15
16        000070  .IVT=     70            ; INTERRUPT VECTOR TABLE ADDRESS
17
18  02400'177777  TTI.:     .TTI          ; CONSOLE INPUT
19  02401'177777  TTO.:     .TTO          ; CONSOLE OUTPUT
20  02402'177777  MI1.:     .MI1          ; MUX. #1 INPUT
21  02403'177777  MO1.:     .MO1          ; MUX. #1 OUTPUT
22  02404'177777  MI2.:     .MI2          ; MUX. #2 INPUT
23  02405'177777  MO2.:     .MO2          ; MUX. #2 OUTPUT
24  02406'177777  MI3.:     .MI3          ; MUX. #3 INPUT
25  02407'177777  MO3.:     .MO3          ; MUX. #3 OUTPUT
26
27  02410'034070  INIT:     LDA     3,.IVT
28  02411'024767              LDA     1,TTI.
29  02412'045410              STA     1,TTI,3
30  02413'024766              LDA     1,TTO.
31  02414'045411              STA     1,TTO,3
32  02415'024765              LDA     1,MI1.
33  02416'045442              STA     1,DMI1,3
34  02417'024764              LDA     1,MO1.
35  02420'045443              STA     1,DMO1,3
36  02421'024763              LDA     1,MI2.
37  02422'045444              STA     1,DMI2,3
38  02423'024762              LDA     1,MO2.
39  02424'045445              STA     1,DMO2,3
40  02425'024761              LDA     1,MI3.
41  02426'045446              STA     1,DMI3,3
42  02427'024760              LDA     1,MO3.
43  02430'045447              STA     1,DMO3,3
```

```
                000064  DAY=    64
                000065  MONTH=  65              ; DINOS DATE WORDS
                000066  YEAR=   66

                        ; INITIALIZE TIME WORDS

02431'034041            LDA 3   DDOS    ; SYSTEM TABLE ADDRESS
02432'021456            LDA 0   D.DAT,3 ; DDOS DATE WORD
02433'024417            LDA 1   .37     ; 5-BIT MASK
02434'107400            AND     0,1     ; EXTRACT 5 BITS
02435'122643            SUBOR   1,0     ; DELETE FROM REST
02436'044064            STA 1   DAY     ; DAY-OF-THE-MONTH

02437'024414            LDA 1   .377    ; HALF-WORD MASK
02440'101300            MOVS    0,0     ; SWAP THE REST
02441'107400            AND     0,1     ; EXTRACT 8 BITS
02442'122743            SUBS    1,0     ; DELETE FROM REST
02443'044066            STA 1   YEAR    ; YEAR SINCE 1900

02444'103120            ADDZL   0,0     ; REST * 4
02445'103120            ADDZL   0,0     ; THEN * 16
02446'135300            MOVS    0,1     ; JUSTIFY
02447'044065            STA 1   MONTH   ; MONTH-OF-THE-YEAR
02450'012451            JMP     @.+1    ; CONTINUE START-UP
02451'000047'           START

02452'000037    .37:    37
02453'000377    .377:   377

        000001  .TXTM   1

02454'040611    FILE:   .TXT    *AIDS.STATUS*
02455'042123
02456'027123
02457'052101
02460'052125
02461'051440
02462'040611    NAME:   .TXT    *AIDS.LOG*
02463'042123
02464'027114
02465'047507
02466'000000

        000011  .BLK    LTIT+SIZE-.     ; REST OF WORKING AREA

02410'  .END    INIT
```


SYMBOL	VALUE	DEF'N	REFERENCES						
AWAIT	'X	2:11	2:39	5:04	5:22	6:04	6:22	7:04	7:22
			8:04	8:22					
BUFF	000045'	3:06	3:16						
CCN	000020	1:21	1:45						
CCTB	$X	2:17	3:23						
CCTE	$X	2:17	3:39						
CHAR	000012	1:14	1:37						
CTIT	000000'	2:36	2:28	2:45	3:03	4:41	11:04		
CZIP	'X	2:09	2:40	9:05	9:23	10:05	10:23		
DATA	000015	1:17	1:39						
DAY	000064	12:01	12:12						
DFLAG	100000	2:24	5:02	5:20	6:02	6:20	7:02	7:20	8:02
			8:20						
DUMP	000046'	3:07	3:25						
DZAP	'X	2:08	5:07	5:25	6:07	6:25	7:07	7:25	8:07
			8:25						
DZIP	'X	2:08	5:05	5:23	6:05	6:23	7:05	7:23	8:05
			8:23						
EDIT	000010	1:12	1:37						
END	000027	1:28	1:50						
FIELD	000016	1:18	1:40						
FILE	002454'	12:32	3:04						
FLAG	000001	1:04	1:35						
FLOG	000044'	3:05	3:15						
FORM	000014	1:16	1:39						
HO	000013	1:15	1:38						
ID	000000	1:03	1:35						
INDEX	000017	1:19	1:40						
INIT	002410'	11:27	12:46						
LEADS	'X	2:13	11:05						
LFLAG	000000	2:26	11:03						
LINE	000036	1:32	1:42	2:45	5:13	5:31	6:13	6:31	7:13
			7:31	8:13	8:31	9:13	9:31	10:13	10:31
LINK	000002	1:05	1:36	4:24					
LOOP	000003	1:06	1:36	4:05					
LTIT	002340'	11:01	2:28	11:11	12:44				
LZAP	'X	2:10	11:08						
LZIP	'X	2:10	11:06						
MI1.	002402'	11:20	11:32						
MI2.	002404'	11:22	11:36						
MI3.	002406'	11:24	11:40						
MO1.	002403'	11:21	11:34						
MO2.	002405'	11:23	11:38						
MO3.	002407'	11:25	11:42						
MODE	000034	1:30	1:41						
MONTH	000065	12:02	12:23						
MORE	000000'	3:24	3:41						
NAME	002462'	12:38	3:05						
NEXT	000101'	4:04	4:27						
OLD	000025	1:26	1:46						
OPT	000021	1:22	1:45						
PAGE	000006	2:21	3:35	4:22					
PADS	000035	1:31	1:41						
PAGE	000037	1:33	1:42						
PFLAG	000004	2:25	2:37	9:02	9:20	10:02	10:20		

SYMBOL	VALUE	DEF'N	REFERENCES						
[illegible]	000012	2:22	3:34	4:21					
[illegible]	'X	2:12	3:07						
[illegible]	'X	2:09	2:42	9:07	9:25	10:07	10:25		
[illegible]	000026	1:27	1:49						
[illegible]	000400'	3:01	3:24	3:33	4:04	4:20			
[illegible]	000140	2:19	4:41	5:17	5:35	6:17	6:35	7:17	7:35
			8:17	8:35	9:17	9:35	10:17	10:35	12:44
[illegible]	'X	2:15	3:06						
[illegible]	000041'	3:02	3:37						
[illegible]	000011	1:13	1:38						
[illegible]	000047'	3:11	12:25						
[illegible]	000043'	3:04	3:11						
[illegible]	000121'	4:24	4:07						
[illegible]	000022	1:23	1:48						
[illegible]	000140'	5:01	2:29	2:38	5:13	5:17			
[illegible]	000300'	5:19	2:29	5:03	5:31	5:35			
[illegible]	000440'	6:01	2:29	5:21	6:13	6:17			
[illegible]	000600'	6:19	2:29	6:03	6:31	6:35			
[illegible]	000740'	7:01	2:30	6:21	7:13	7:17			
[illegible]	001100'	7:19	2:30	7:03	7:31	7:35			
[illegible]	001240'	8:01	2:30	7:21	8:13	8:17			
[illegible]	001400'	8:19	2:30	8:03	8:31	8:35			
[illegible]	001540'	9:01	2:31	8:21	9:13	9:17			
[illegible]	001700'	9:19	2:31	9:03	9:31	9:35			
[illegible]	002040'	10:01	2:31	9:21	10:13	10:17			
[illegible]	002200'	10:19	2:31	10:03	10:31	10:35			
[illegible]	002340'	11:02	10:21						
[illegible]	000042'	3:03	4:03	4:25					
[illegible]	000024	1:25	1:46						
[illegible]	002400'	11:18	11:28						
[illegible]	002401'	11:19	11:30						
[illegible]	'X	2:07	2:41	5:06	5:24	6:06	6:24	7:06	7:24
			8:06	8:24	9:06	9:24	10:06	10:24	11:07
[illegible]	'X	2:07	2:43	5:08	5:26	6:08	6:26	7:08	7:26
			8:08	8:26	9:08	9:26	10:08	10:26	11:09
[illegible]	000023	1:24	1:47						
[illegible]	'X	2:14	4:30						
[illegible]	000066	12:03	12:13						
[illegible]	000006	1:09	1:44	4:13					
[illegible]	000005	1:08	1:43						
[illegible]	000004	1:07	1:43						
[illegible]	000007	1:10	1:44						
[illegible]	002452'	12:27	12:09						
[illegible]	002453'	12:28	12:14						
[illegible]	000127'	4:32	3:28	4:15					
[illegible]	000070	11:16	11:27						
[illegible]	'X	2:04	11:20						
[illegible]	'X	2:05	11:22						
[illegible]	'X	2:06	11:24						
[illegible]	'X	2:04	11:21						
	'X	2:05	11:23						
	'X	2:06	11:25						
[illegible]	000135'	4:39	4:12						
[illegible]	000132'	4:35	3:27	3:32	4:09	4:19			
[illegible]	'X	2:03	11:13						

SYMBOL	VALUE	DEF'N	REFERENCES
[illegible]	'X	2:03	11:19

FLAGGED LINES: NONE

TERMINAL INPUT AND OUTPUT (TRIO)

The terminal input and output routine does all the busy work of controlling the multiplexor channels for input and output to the terminals.


```
                        .HEAD   A I D S      -       TERMINAL INPUT AND OUTPUT
                        .TITLE  TRIO
                        .EXTD   ID                  ; TERMINAL NUMBER
              000055    RZEP=   R5                  ; RETURN
              000057    RZOP=   R7                  ; WORDS
                        .ENT    .TTI,.TTO
                        .ENT    .MI1,.MO1           ; INTERRUPT
                        .ENT    .MI2,.MO2           ; ENTRIES
                        .ENT    .MI3,.MO3
                        .ENT    TZEP,TZOP           ; DRIVER ENTRIES
                        .NREL
                        ; TERMINAL INPUT
00000'055055  TZEP:     STA     3,RZEP,2; SAVE RETURN
00001'000402            JMP     ZEP2    ; JUMP AROUND
00002'006040  ZEP1:     IDLE            ; WAIT AWHILE
00003'021001$ ZEP2:     LDA     0,ID,2  ; TERMINAL NUMBER
00004'103120            ADDZL   0,0     ; TIMES FOUR
00005'034410            LDA     3,.MXT  ; TERMINAL TABLE
00006'117000            ADD     0,3     ; ENTRY ADDRESS
00007'025402            LDA     1,NEW,3 ; NEXT CHARACTER
00010'124015            COM#    1,1,SNR ; IF NOT PRESENT,
00011'000771            JMP     ZEP1    ; GO BACK AND WAIT
00012'102000            ADC     0,0
00013'041402            STA     0,NEW,3 ; SET FLAG
00014'003055            JMP     @RZEP,2 ; AND EXIT
00015'000140' .MXT:     MXT             ; TERMINAL TABLE
```

```
                        ; TERMINAL OUTPUT

00016'055057  TZOP:   STA     3,RZOP,2; SAVE RETURN
00017'000402          JMP     ZOP2    ; JUMP AROUND
00020'000000  ZOP1:   IDLE            ; WAIT AWHILE

00021'021001$ ZOP2:   LDA     0,ID,2  ; TERMINAL NUMBER
00022'133123          ADDZL   0,0     ; TIMES FOUR
00023'034772          LDA     3,.MXT  ; TERMINAL TABLE
00024'117000          ADD     0,3     ; ENTRY ADDRESS
00025'011403          ISZ     OLD,3   ; IF NOT YET READY,
00026'000772          JMP     ZOP1    ; GO BACK AND WAIT

00027'020412          LDA     0,WAIT  ; GET TIMEOUT
00030'041403          STA     0,OLD,3 ; SET COUNTER
00031'021401          LDA     0,INST,3; OUTPUT INSTRUCTION
00032'040035          STA     0,ZOP3  ; PUT IT IN-LINE
00033'021401          LDA     0,UNIT,3; UNIT NUMBER

00034'034502          LDA     3,LO    ; LO-BYTE MASK
00035'137400          AND     1,3     ; EXTRACT DATA
00036'163003          ADD     3,0     ; ADD THE UNIT
00037'061111  ZOP3:   DOAS    0,TTO   ; OUTPUT IT
00040'000357          JMP     @RZOP,2 ; AND EXIT

00041'174000  WAIT:   -4000           ; TIMEOUT COUNT
```



```
                        ; INTERRUPT ENTRIES

00042'030413  .MI1:   LDA     2,.MX1  ; ENTRY TABLE
00043'061442          DIB     0,DMI1  ; READ STATUS
00044'004442          JSR     MIO     ; PROCESS
00045'062042          DOB     0,DMI1  ; SELECT UNIT
00046'060642          DIAC    0,DMI1  ; READ DATA
00047'000451          JMP     MI      ; FINISH

00050'030405  .MO1:   LDA     2,.MX1  ; ENTRY TABLE
00051'061443          DIB     0,DMO1  ; READ STATUS
00052'004434          JSR     MIO     ; PROCESS
00053'062043          DOB     0,DMO1  ; CLEAR FLAG
00054'000450          JMP     MO      ; FINISH

00055'000200' .MX1:   MX1             ; MUX. 1 ENTRIES

00056'030413  .MI2:   LDA     2,.MX2  ; ENTRY TABLE
00057'061444          DIB     0,DMI2  ; READ STATUS
00060'004426          JSR     MIO     ; PROCESS
00061'062044          DOB     0,DMI2  ; SELECT UNIT
00062'060644          DIAC    0,DMI2  ; READ DATA
00063'000435          JMP     MI      ; FINISH

00064'030405  .MO2:   LDA     2,.MX2  ; ENTRY TABLE
00065'061445          DIB     0,DMO2  ; READ STATUS
00066'004420          JSR     MIO     ; PROCESS
00067'062045          DOB     0,DMO2  ; CLEAR FLAG
00070'000434          JMP     MO      ; FINISH

00071'000240' .MX2:   MX2             ; MUX. 2 ENTRIES

00072'030413  .MI3:   LDA     2,.MX3  ; ENTRY TABLE
00073'061446          DIB     0,DMI3  ; READ STATUS
00074'004412          JSR     MIO     ; PROCESS
00075'062046          DOB     0,DMI3  ; SELECT UNIT
00076'060646          DIAC    0,DMI3  ; READ DATA
00077'000421          JMP     MI      ; FINISH

00100'030405  .MO3:   LDA     2,.MX3  ; ENTRY TABLE
00101'061447          DIB     0,DMO3  ; READ STATUS
00102'004404          JSR     MIO     ; PROCESS
00103'062047          DOB     0,DMO3  ; CLEAR FLAG
00104'000420          JMP     MO      ; FINISH

00105'000300' .MX3:   MX3             ; MUX. 3 ENTRIES
```

```
00106'024431  MIO:    LDA     1,HI      ; HI-BYTE MASK
00107'123400          AND     1,0       ; GET READY BITS
00110'126401          SUB     1,1,SKP   ; START AT ZERO

00111'125400          INC     1,1       ; BUMP UNIT NUMBER
00112'101123          MOVZL   0,0,SNC   ; TEST STATUS BIT
00113'000776          JMP     .-2       ; IF ZERO, REPEAT

00114'121300          MOVS    1,0       ; UNIT TO HI-BYTE
00115'127120          ADDZL   1,1       ; UNIT TIMES FOUR
00116'133000          ADD     1,2       ; ENTRY ADDRESS
00117'001400          JMP     0,3       ; RETURN

00120'024416  MI:     LDA     1,LO      ; LO-BYTE MASK
00121'123400          AND     1,0       ; EXTRACT DATA
00122'041002          STA     0,NEW,2   ; AND SAVE IT
00123'002004          INTR              ; THAT'S ALL

00124'126000  MO:     ADC     1,1       ; ALL-ONES
00125'041003          STA     1,OLD,2   ; SET ALL BITS
00126'002004          INTR              ; THAT'S ALL

00127'060610  .TTI:   DIAC    0,TTI     ; READ AND
00130'040432          STA     0,TTNEW   ; SAVE DATA
00131'002004          INTR              ; RETURN

00132'060211  .TTO:   NIOC    TTO       ; CLEAR FLAG
00133'126000          ADC     1,1
00134'044427          STA     1,TTOLD   ; SET ALL BITS
00135'002004          INTR              ; RETURN

00136'000377  LO:     377               ; LO-BYTE MASK
00137'177400  HI:     377*400           ; HI-BYTE MASK
```



```
                      ; TERMINAL ENTRY TABLE

              000000  UNIT=   0                 ; UNIT NUMBER
              000001  INST=   1                 ; START INST.
              000002  NEW=    2                 ; NEXT INPUT
              000003  OLD=    3                 ; LAST OUTPUT

00140'000020  MXT:    .REP    16.             ; ENTRIES
00140'000000          0                       ; 0,1,2,3

00160'000000          0                       ; TT UNIT = 0
00161'061111          DOAS    0,TTO           ; START INST
00162'177777  TTNEW:  -1                      ; NEXT INPUT
00163'177777  TTOLD:  -1                      ; LAST OUTPUT

              000014          .REP    12.     ; ENTRIES
00164'000000          0                       ; 5,6,7
```

```
                                                0*400   ; UNIT 0
00241'061043            DOA     0,DMO1  ; MUX. 1
00242'177777            -1              ; INPUT
00243'177777            -1              ; OUTPUT

                                                1*400   ; UNIT 1
00245'061043            DOA     0,DMO1  ; MUX. 1
00246'177777            -1              ; INPUT
00247'177777            -1              ; OUTPUT

00250'001000                            2*400   ; UNIT 2
00251'061043            DOA     0,DMO1  ; MUX. 1
00252'177777            -1              ; INPUT
00253'177777            -1              ; OUTPUT

00254'001400                            3*400   ; UNIT 3
00255'061043            DOA     0,DMO1  ; MUX. 1
00256'177777            -1              ; INPUT
00257'177777            -1              ; OUTPUT

00260'002000                            4*400   ; UNIT 4
00261'061043            DOA     0,DMO1  ; MUX. 1
00262'177777            -1              ; INPUT
00263'177777            -1              ; OUTPUT

00264'002400                            5*400   ; UNIT 5
00265'061043            DOA     0,DMO1  ; MUX. 1
00266'177777            -1              ; INPUT
00267'177777            -1              ; OUTPUT

00270'003000                            6*400   ; UNIT 6
00271'061043            DOA     0,DMO1  ; MUX. 1
00272'177777            -1              ; INPUT
00273'177777            -1              ; OUTPUT

00274'003400                            7*400   ; UNIT 7
00275'061043            DOA     0,DMO1  ; MUX. 1
00276'177777            -1              ; INPUT
00277'177777            -1              ; OUTPUT
```



```
 1  00240'000000  MX2:                      0*400   ; UNIT 0
 2  00241'061045            DOA     0,DMO2  ; MUX. 2
 3  00242'177777            -1              ; INPUT
 4  00243'177777            -1              ; OUTPUT
 5
 6  00244'000400                            1*400   ; UNIT 1
 7  00245'061045            DOA     0,DMO2  ; MUX. 2
 8  00246'177777            -1              ; INPUT
 9  00247'177777            -1              ; OUTPUT
10
11  00250'001000                            2*400   ; UNIT 2
12  00251'061045            DOA     0,DMO2  ; MUX. 2
13  00252'177777            -1              ; INPUT
14  00253'177777            -1              ; OUTPUT
15
16  00254'001400                            3*400   ; UNIT 3
17  00255'061045            DOA     0,DMO2  ; MUX. 2
18  00256'177777            -1              ; INPUT
19  00257'177777            -1              ; OUTPUT
20
21  00260'002000                            4*400   ; UNIT 4
22  00261'061045            DOA     0,DMO2  ; MUX. 2
23  00262'177777            -1              ; INPUT
24  00263'177777            -1              ; OUTPUT
25
26  00264'002400                            5*400   ; UNIT 5
27  00265'061045            DOA     0,DMO2  ; MUX. 2
28  00266'177777            -1              ; INPUT
29  00267'177777            -1              ; OUTPUT
30
31  00270'003000                            6*400   ; UNIT 6
32  00271'061045            DOA     0,DMO2  ; MUX. 2
33  00272'177777            -1              ; INPUT
34  00273'177777            -1              ; OUTPUT
35
36  00274'003400                            7*400   ; UNIT 7
37  00275'061045            DOA     0,DMO2  ; MUX. 2
38  00276'177777            -1              ; INPUT
39  00277'177777            -1              ; OUTPUT
```

```
000300'000000  MX3:             0*400    ; UNIT 0
000301'061047                DOA 0,DMO3   ; MUX. 3
000302'177777                -1           ; INPUT
000303'177777                -1           ; OUTPUT

000304'000400                1+400    ; UNIT 1
000305'061047                DOA 0,DMO3   ; MUX. 3
000306'177777                -1           ; INPUT
000307'177777                -1           ; OUTPUT

000310'001000                2*400    ; UNIT 2
000311'061047                DOA 0,DMO3   ; MUX. 3
000312'177777                -1           ; INPUT
000313'177777                -1           ; OUTPUT

000314'001400                3*400    ; UNIT 3
000315'061047                DOA 0,DMO3   ; MUX. 3
000316'177777                -1           ; INPUT
000317'177777                -1           ; OUTPUT

000320'002000                4*400    ; UNIT 4
000321'061047                DOA 0,DMO3   ; MUX. 3
000322'177777                -1           ; INPUT
000323'177777                -1           ; OUTPUT

000324'002400                5*400    ; UNIT 5
000325'061047                DOA 0,DMO3   ; MUX. 3
000326'177777                -1           ; INPUT
000327'177777                -1           ; OUTPUT

000330'003000                6*400    ; UNIT 6
000331'061047                DOA 0,DMO3   ; MUX. 3
000332'177777                -1           ; INPUT
000333'177777                -1           ; OUTPUT

000334'003400                7*400    ; UNIT 7
000335'061047                DOA 0,DMO3   ; MUX. 3
000336'177777                -1           ; INPUT
000337'177777                -1           ; OUTPUT

                .END
```


SYMBOL	VALUE	DEF'N	REFERENCES					
HI	000137'	4:33	4:01					
ID	$X	1:05	1:25	2:07				
INST	000001	5:04	2:16					
LO	000136'	4:32	2:20	4:14				
MI	000120'	4:14	3:08	3:23	3:38			
MIO	000106'	4:01	3:05	3:12	3:20	3:27	3:35	3:42
MO	000124'	4:19	3:14	3:29	3:44			
MX1	000200'	6:01	3:16					
MX2	000240'	7:01	3:31					
MX3	000300'	8:01	3:46					
MXT	000140'	5:08	1:38					
NEW	000002	5:05	1:30	1:35	4:16			
OLD	000003	5:06	2:11	2:15	4:20			
RZEP	000055	1:07	1:21	1:36				
RZOP	000057	1:08	2:03	2:24				
TTNEW	000162'	5:13	4:24					
TTOLD	000163'	5:14	4:29					
TZEP	000000'	1:21	1:15					
TZOP	000016'	2:03	1:15					
UNIT	000003	5:03	2:18					
WAIT	000041'	2:26	2:14					
ZEP1	000002'	1:23	1:32					
ZEP2	000003'	1:25	1:22					
ZOP1	000020'	2:05	2:12					
ZOP2	000021'	2:07	2:04					
ZO	000037'	2:23	2:17					
.MI1	000042'	3:03	1:11					
.MI2	000056'	3:18	1:12					
.MI3	000072'	3:33	1:13					
.MO1	000050'	3:10	1:11					
.MO2	000064'	3:25	1:12					
.MO3	000100'	3:40	1:13					
.MX1	000055'	3:16	3:03	3:10				
.MX2	000071'	3:31	3:18	3:25				
.MX3	000105'	3:46	3:33	3:40				
.MXT	000015'	1:38	1:27	2:09				
.TTI	000127'	4:23	1:10					
.TTO	000132'	4:27	1:10					

FLAGGED LINES: NONE

CONTROL CONSOLE INPUT (CZIP)

The control console input routine gets an input character from the console input device through a call to ZEP, the keyboard input handler. It performs a translation on control codes and the rubout code and returns the input character in AC1.

```
PAGE  1        A I D S      -       CONTROL CONSOLE INPUT
02/20/74
                        .HEAD   A I D S      -       CONTROL CONSOLE INPUT

                        .TITLE  CZIP

                        .EXTD   ZEP     ; INPUT DRIVER

            000054      RZIP=   R4      ; RETURN WORD

                        .ENT    CZIP

                        .NREL

00000'055054    CZIP:   STA     3,RZIP,2        ; SAVE RETURN
00001'007001$           JSR     @ZEP,2          ; GET INPUT

00002'020413            LDA     0,.177          ; 7-BIT MASK
00003'107400            AND     0,1             ; EXTRACT DATA
00004'106415            SNE     0,1             ; IF RUBOUT,
00005'126000            ADC     1,1             ; SUBSTITUTE

00006'020410            LDA     0,.40           ; CONTROL LIMIT
00007'106513            SGT     0,1             ; IF PRINTABLE,
00010'003054            JMP     @RZIP,2         ; RETURN

00011'004401            JSR     ZOT             ; SELF-LOCATE
00012'137000    ZOT:    ADD     1,3             ; COMPUTE ADDRESS
00013'025406            LDA     1,ZIT-ZOT,3     ; AND TRANSLATE
00014'003054            JMP     @RZIP,2         ; RETURN

00015'000177    .177:   177     ; 7-BIT MASK
00016'000040    .40:    40      ; ASCII SPACE
```

```
                ; TRANSLATION TABLE

                                00      ;177 = RUBOUT
000020'         ZIT:
                                20      ; 00 = NULL
                                00
                                00
                                00
                                00
                                00
                                00
                                00

                                30      ; 10 = BACKSPACE
                                31      ; 11 = TAB
                                27      ; 12 = END
                                00
                                17      ; 14 = EJECT
                                35      ; 15 = RETURN
                                00
                                00

                                00
                                00
                                00
                                00
                                00
                                00
                                00

                                00
                                00
                                00
                                01      ; 33 = ESC
                                00
                                00
                                00
                                00

                .END
```



```
SYMBOL  VALUE    DEF'N   REFERENCES

CZIP    000000'   1:13    1:09
RZIP    000054    1:07    1:13    1:23    1:28
ZEP         $X    1:05    1:14
ZIT     000020'   2:04    1:27
ZOT     000012'   1:26    1:25    1:27
.177    000015'   1:30    1:16
.40     000016'   1:31    1:21

        FLAGGED LINES: NONE
```

DISPATCH CONSOLE INPUT (DZIP)

The dispatch console input routine gets an input character from the terminal input device through a call to ZEP, the keyboard input handler. It performs a translation (differing from the translation done by CZIP) on control codes and returns the input character in AC1.


```
                        .HEAD   A I D S      -       DISPATCH CONSOLE INPUT
                        .TITLE  DZIP
                        .EXTD   ZEP     ; INPUT DRIVER
          000054    RZIP=   R4      ; RETURN WORD
                        .ENT    DZIP
                        .NREL
00000'055054  DZIP:     STA     3,RZIP,2        ; SAVE RETURN
00001'007001$           JSR     @ZEP,2          ; GET INPUT
00002'020413            LDA     0,.177          ; 7-BIT MASK
00003'107400            AND     0,1             ; EXTRACT DATA
00004'106415            SNE     0,1             ; IF RUBOUT,
00005'024412            LDA     1,.26           ; SUBSTITUTE
00006'020410            LDA     0,.40           ; CONTROL LIMIT
00007'106513            SGT     0,1             ; IF PRINTABLE,
00010'003054            JMP     @RZIP,2         ; RETURN
00011'004401            JSR     ZOT             ; SELF-LOCATE
00012'137000  ZOT:      ADD     1,3             ; COMPUTE ADDRESS
00013'025406            LDA     1,ZIT-ZOT,3     ; AND TRANSLATE
00014'003054            JMP     @RZIP,2         ; RETURN
00015'000177  .177:     177     ; 7-BIT MASK
00016'000040  .40:      40      ; ASCII SPACE
00017'000026  .26:      26      ; SUBSTITUTION
```

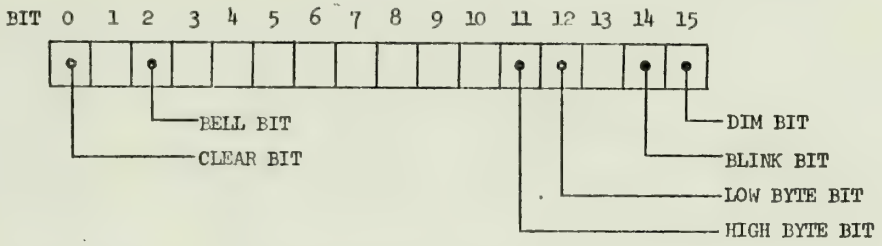
```
                ; TRANSLATION TABLE

000020'         ZIT:
000020'000014           14      ; KEY 14 PRODUCES 00
000021'000002           02      ; KEY 02 PRODUCES 01
000022'000004           04      ; KEY 04 PRODUCES 02
000023'000011           11      ; KEY 11 PRODUCES 03
000024'000013           13      ; KEY 13 PRODUCES 04
000025'000006           06      ; KEY 06 PRODUCES 05
000026'000000           00
000027'000015           15      ; KEY 15 PRODUCES 07

000030'000030           30      ; KEY 30 PRODUCES 10 (BS)
000031'000031           31      ; KEY 31 PRODUCES 11 (TAB)
000032'000027           27      ; KEY 27 PRODUCES 12 (END)
000033'000037           37      ; KEY 37 PRODUCES 13 (INS)
000034'000017           17      ; KEY 17 PRODUCES 14
000035'000035           35      ; KEY 35 PRODUCES 15 (RET)
000036'000000           00
000037'000000           00

000040'000010           10      ; KEY 10 PRODUCES 20
000041'000000           00
000042'000000           00
000043'000000           00
000044'000000           00
000045'000000           00
000046'000007           07      ; KEY 07 PRODUCES 26 <- 177
000047'000012           12      ; KEY 12 PRODUCES 27

000050'000003           03      ; KEY 03 PRODUCES 30
000051'000005           05      ; KEY 05 PRODUCES 31
000052'000016           16      ; KEY 16 PRODUCES 32
000053'000001           01      ; KEY 01 PRODUCES 33
000054'000026           26      ; KEY 26 PRODUCES 34 (TOP)
000055'000032           32      ; KEY 32 PRODUCES 35 (REV)
000056'000033           33      ; KEY 33 PRODUCES 36 (ADV)
000057'000036           36      ; KEY 36 PRODUCES 37 (DEL)

                .END
```


SYMBOL	VALUE	DEF'N	REFERENCES		
DZIP	000000'	1:13	1:09		
RZIP	000054	1:07	1:13	1:23	1:28
ZEP	$X	1:05	1:14		
ZIT	000020'	2:03	1:27		
ZOT	000012'	1:26	1:25	1:27	
.177	000015'	1:30	1:16		
.26	000017'	1:32	1:19		
.40	000016'	1:31	1:21		

FLAGGED LINES: NONE

ADDS CRT DISPLAY OUTIUT (DZAP)

The ADDS CRT display output routine is used to output to an ADDS CRT display terminal. It receives its input characters in AC1 and a flag word in AC0. The action taken due to the flag word is as follows:



If the clear bit is high, the routine clears the screen and repositions the cursor to the upper left-hand corner of the screen.

If the bell bit is high, the routine sends a bell code.

If the high byte bit is high, the routine sends the character in the upper byte of the data word (AC1).

If the low byte bit is high, the routine sends the character in the lower byte of the data word.

If both bits are high, both bytes are sent in the above order.

If the dim bit is high, the data is sent "tagged" (reduced intensity).

If the blink bit is high, the data is made to blink.

If neither the high byte nor low byte bits are high, the input data word is taken to be a screen address and the cursor is moved to the position indicated. The upper byte is taken as a line number and the lower byte as the column number within the line.

```
PAGE  1          A I D S      -       ADDS CRT DISPLAY OUTPUT
02/20/74
                 .HEAD   A I D S      -       ADDS CRT DISPLAY OUTPUT

                 .TITLE  DZAP

                 .EXTD   LINE     ; LINE LENGTH
                 .EXTD   PAGE     ; PAGE SIZE
                 .EXTD   ZOP      ; OUTPUT DRIVER
                 .EXTD   MODE     ; DISPLAY MODE WORD
                 .EXTD   SPOT     ; CURSOR POSITION
                 .EXTD   HOME     ; OFFICIAL POSITION

        000056   RZAP=   R6       ; RETURN WORD

        000072   BITS=   T2       ; OPTION BITS
        000073   DATA=   T3       ; DATA / POSITION
        000074   MOVER=  T4       ; SUBROUTINE RETURN ADDRESS
        000075   TEMP=   T5       ; TEMPORARY STORAGE
        000077   SAVE=   T7       ; TEMPORARY SAVE WORD

                 .ENT    DZAP
                 .NREL

00000'055056     DZAP:   STA 3    RZAP,2  ; SAVE RETURN
00001'045073             STA 1    DATA,2  ; SAVE DATA
00002'041072             STA 0    BITS,2  ; AND FLAGS

00003'101113             MOVL#    0,0 SNC ; TEST CLEAR BIT
00004'000413             JMP      OK      ; IF NOT SET, GO

00005'024571             LDA 1    .14     ; GET ERASE
00006'004565             JSR      SEND    ; SEND IT
00007'024570             LDA 1    .36     ; GET FORMAT-ON
00010'004563             JSR      SEND    ; SEND IT
00011'024567             LDA 1    .17     ; GET STOP-TAG
00012'004561             JSR      SEND    ; SEND IT

00013'126400             SUB      1,1     ; MAKE A ZERO
00014'045005$            STA 1    SPOT,2  ; REPOSITION
00015'045006$            STA 1    HOME,2  ; AND UPDATE
00016'045004$            STA 1    MODE,2  ; CLEAR TAG AND BLINK BITS

00017'021072     OK:     LDA 0    BITS,2  ; GET FLAG WORD
00020'101222             MOVZR    0,0 SZC ; TEST TAG BIT
00021'000413             JMP      TON     ; IF SET, START

00022'025004$            LDA 1    MODE,2  ; GET TAG STATE
00023'125233             MOVZR#   1,1 SNC ; AND CHECK IT
00024'000421             JMP      TOK     ; IF CLEAR, OK

00025'024553             LDA 1    .17     ; GET STOP-TAG
00026'004545             JSR      SEND    ; SEND IT
00027'025004$            LDA 1    MODE,2  ; FETCH MODE WORD
00030'125200             MOVR     1,1     ; REMOVE TAG BIT
00031'125120             MOVZL    1,1     ; SET TAG BIT
00032'045004$            STA 1    MODE,2  ; RESET MODE WORD
00033'000412             JMP      TOK

00034'025004$    TON:    LDA 1    MODE,2  ; GET TAG STATE
```

```
00035'125232            MOVZR#  1,1 SZC ; AND CHECK IT
00036'000407            JMP     TOK     ; IF SET, OK

00037'024551            LDA 1   .16     ; GET START-TAG
00040'004533            JSR     SEND    ; SEND IT
00041'025004$           LDA 1   MODE,2  ; GET MODE WORD
00042'125200            MOVR    1,1     ; REMOVE TAG BIT
00043'125140            MOVOL   1,1     ; SET TAG BIT
00044'045004$           STA 1   MODE,2  ; RESET MODE WORD

00045'021072  TOK:      LDA 0   BITS,2  ; GET FLAG WORD
00046'101220            MOVZR   0,0     ; IGNORE TAG BIT
00047'101222            MOVZR   0,0 SZC ; TEST BLINK BIT
00050'000415            JMP     BON     ; IF SET, GO

00051'041075            STA 0   TEMP,2  ; SAVE FLAGS
00052'025004$           LDA 1   MODE,2  ; GET MODE WORD
   53'125200            MOVR    1,1     ; ROTATE
00054'125243            MOVR    1,1 SNC ; TEST BLINK BIT
   55'000424            JMP     DING    ; IF CLEAR, OK

   56'024534            LDA 1   .40     ; GET STOP-BLINK
00057'004514            JSR     SEND    ; SEND IT
00060'025004$           LDA 1   MODE,2  ; GET MODE WORD
00061'034514            LDA 3   MASK    ; FETCH MASK
00062'167403            AND     3,1     ; CLEAR BLINK BIT
   63'045004$           STA 1   MODE,2  ; RESET MODE WORD
   64'000415            JMP     DING

   65'041075  BON:      STA 0   TEMP,2  ; SAVE FLAGS
   66'025004$           LDA 1   MODE,2  ; GET MODE WORD
00067'125200            MOVR    1,1     ; ROTATE
00070'125202            MOVR    1,1 SZC ; TEST BLINK BIT
00071'000410            JMP     DING    ; IF SET, OK

00072'024517            LDA 1   .34     ; GET START-BLINK
00073'004500            JSR     SEND    ; SEND IT
   74'025004$           LDA 1   MODE,2  ; GET MODE WORD
00075'034503            LDA 3   MASK    ; FETCH MASK
00076'167400            AND     3,1     ; REMOVE BLINK BIT
00077'166003            ADC     3,1     ; SET BLINK BIT
00100'045004$           STA 1   MODE,2  ; RESET MODE WORD

00101'021072  DING:     LDA 0   BITS,2  ; FETCH FLAG WORD
   02'101001            MOVL    0,0     ; IGNORE CLEAR BIT
00103'101003            MOVL    0,0     ; IGNORE MOTOR-OFF BIT
00104'101003            MOVL    0,0 SNC ; TEST BELL FLAG
00105'000403            JMP     BOK     ; NO BELL WANTED
00106'024475            LDA 1   .7      ; GET ASCII BELL CODE
00107'004464            JSR     SEND    ; SEND IT

00110'021075  BOK:      LDA 0   TEMP,2  ; GET FLAGS
00111'024472            LDA 1   .7      ; 3-BIT MASK
   12'123625            ANDZR   1,0 SNR ; GET MODE BITS
00113'000414            JMP     MOVE    ; IF ZERO, GO
   14'025073            LDA 1   DATA,2  ; GET DATA
00115'125303            MOVS    1,1     ; ROTATE BYTES
   16'101236            MOVZR#  0,0 SEZ ; BOTH BYTES?
```



```
00117'000404            JMP     BOTH    ; YES, SEND BOTH
00120'101234            MOVZR#  0,0 SZR ; WHICH OF THE TWO?
00121'000404            JMP     HIGH    ; SEND THE HIGH BYTE
00122'000402            JMP     LOW     ; SEND THE LOW BYTE
00123'004424  BOTH:     JSR     TYPE    ; SEND A BYTE
00124'025073  LOW:      LDA 1   DATA,2  ; RECOVER DATA
00125'004422  HIGH:     JSR     TYPE    ; SEND A BYTE
00126'000402            JMP     ALL     ; RETURN

00127'004464  MOVE:     JSR     MOVES   ; REPOSITION CURSOR

00130'021072  ALL:      LDA 0   BITS,2  ; RESTORE FLAGS
00131'024450            LDA 1   .100    ; GET UPDATE MASK
00132'107405            AND     0,1 SNR ; UPDATE POSITION?
00133'000403            JMP     .+3     ; NO, GO ON
00134'025005$           LDA 1   SPOT,2  ; GET NEW POSITION
00135'045006$           STA 1   HOME,2  ; UPDATE POSITION

00136'103120            ADDZL   0,0     ; IGNORE TOP BITS
00137'103123            ADDZL   0,0 SNC ; TEST "LINE CLEAR" BIT
00140'000404            JMP     FINI    ; OFF, JUST LEAVE

00141'024446            LDA 1   .15     ; ASCII "CR"
00142'004431            JSR     SEND    ; CLEAR REST OF LINE
00143'004461            JSR     REPO    ; REPOSITION

00144'021072  FINI:     LDA 0   BITS,2  ; RESTORE BITS
00145'025073            LDA 1   DATA,2  ; RESTORE DATA/POSITION
00146'043056            JMP     @RZAP,2 ; RETURN

00147'020435  TYPE:     LDA 0   .177    ; FETCH MASK
00150'107400            AND     0,1     ; EXTRACT DATA CHARACTER
00151'020431            LDA 0   .37     ; HIGHEST CONTROL CODE
00152'122513            SGT     1,0     ; TEST FOR CONTROL CHARACTER
00153'000512            JMP     CNTRL   ; IS CONTROL
00154'045077            STA 1   SAVE,2  ; SAVE DATA
00155'020427            LDA 0   .177    ; FETCH MASK
00156'025005$           LDA 1   SPOT,2  ; GET POSITION
00157'107400            AND     0,1     ; GET POSITION IN LINE
00160'021001$           LDA 0   LINE,2  ; COLUMN LIMIT
00161'122112            SLT     1,0     ; PAST END OF LINE?
00162'001400            JMP     0,3     ; YES, RETURN
00163'021005$           LDA 0   SPOT,2  ; GET POSITION
00164'024410            LDA 1   HMASK   ; GET HI-BYTE MASK
00165'107700            ANDS    0,1     ; EXTRACT LINE NUMBER
00166'021002$           LDA 0   PAGE,2  ; LINE LIMIT
00167'122112            SLT     1,0     ; PAST END OF PAGE?
00170'001400            JMP     0,3     ; YES, RETURN
00171'025077            LDA 1   SAVE,2  ; RESTORE DATA
00172'011005$           ISZ     SPOT,2  ; BUMP POSITION
00173'003003$ SEND:     JMP     @ZOP,2  ; SEND THE DATA

00174'177400  HMASK:    177400          ; HI-BYTE MASK
00175'177775  MASK:     177775          ; BLINK BIT MASK

00176'000014  .14:      14              ; FF = SCREEN ERASE
00177'000036  .36:      36              ; RS = FORMAT ON
00200'000017  .17:      17              ; SI = STOP TAG
```

```
00241'000100  .100:   100             ; BIT 9 IS UPDATE BIT
00242'000037  .37:    37              ; US = GORMAT OFF
00243'000007  .7:     7               ; 3-BIT MASK
00244'000177  .177:   177             ; 7-BIT MASK
00245'000012  .10.:   10.             ; DECIMAL TEN
00246'000013  .13:    13              ; VT = LINE ADDRESS
00247'000015  .15:    15              ; ASCII "CR"
00250'000016  .16:    16              ; SO = START TAG
00251'000134  .34:    134             ; START BLINK IS "\"
00252'000040  .40:    40              ; STOP BLINK IS "SPACE"

00253'021073  MOVES:  LDA 0   DATA,2  ; GET NEW POSITION
00254'025005$         LDA 1   SPOT,2  ; AND OLD POSITION
00255'041005$         STA 0   SPOT,2  ; UPDATE POSITION
00256'122405          SUB     1,0 SNR ; IF EQUAL,
00257'001400          JMP     0,3     ; DON'T BOTHER
00260'024764          LDA 1   .177    ; FETCH MASK
00261'147400          AND     0,1     ; EXTRACT # OF COLUMNS TO MOVE
00262'106415          SNE     0,1     ; POSITIVE MOVE ON THIS LINE?
00263'000414          JMP     SCOL    ; YES, JUST MOVE COLUMNWISE

00264'056074  REPO:   STA 3   MOVER,2 ; SAVE RETURN ADDRESS
00265'024761          LDA 1   .13     ; GET LINE-SELECT
00266'034745          JSR     SEND    ; SEND IT
00267'024433          LDA 1   AA      ; ALPHA BITS
00270'021005$         LDA 0   SPOT,2  ; GET POSITION
00271'107300          ADDS    0,1     ; GET LINE NUMBER
00272'034741          JSR     SEND    ; SEND IT

00273'021005$         LDA 0   SPOT,2  ; GET POSITION
00274'024754          LDA 1   .177    ; 7-BIT MASK
00275'123405          AND     1,0 SNR ; IF COLUMN ZERO,
00276'003074          JMP     @MOVER,2        ; DONE

00277'041075  SCOL:   STA 0   TEMP,2  ; SAVE COLUMN NUMBER
00300'024424          LDA 1   EE      ; GET ESC/ENQ
00301'034732          JSR     SEND    ; SEND ESC
00302'125300          MOVS    1,1     ; SWAP BYTES
00303'034733          JSR     SEND    ; SEND ENQ
00304'021075          LDA 0   TEMP,2  ; GET COLUMN NUMBER

00305'176000          ADC     3,3     ; MAKE MINUS ONE
00306'024737          LDA 1   .10.    ; DECIMAL TEN
00307'175400          INC     3,3     ; BUMP TENS DIGIT
00310'122443          SUBO    1,0 SNC ; SUBTRACT TEN
00311'000776          JMP     .-2     ; REPEAT UNTIL OFLOW
00312'123300          ADDS    1,0     ; RESTORE AND SWAP
00313'163000          ADD     3,0     ; INSERT TENS DIGIT

00314'024407          LDA 1   ZZ      ; NUMERIC BITS
00315'117000          ADD     0,1     ; MAKE NUMERIC
00316'034715          JSR     SEND    ; SEND TENS DIGIT
00317'125300          MOVS    1,1     ; SWAP BYTES
00320'034713          JSR     SEND    ; SEND ONES DIGIT
00321'003074          JMP     @MOVER,2        ; RETURN

00322'040100  AA:     40100           ; ALPHA BITS
00323'030060  ZZ:     30060           ; NUMERIC BITS
```



```
00264'002433  EE:     2433            ; ESC/ENQ

00265'020716  CNTRL:  LDA 0   .7      ; .7 = BELL
00266'106415          SNE     0,1     ; TEST FOR BELL
00267'000704          JMP     SEND    ; IS A BELL, SEND IT

00270'020502          LDA 0   .10     ; .10 = BACK SPACE
00271'106414          SEQ     0,1     ; TEST FOR BACK SPACE
00272'000410          JMP     THT     ; NOT, TEST FOR HOR. TAB
00273'021005$         LDA 0   SPOT,2  ; PICK UP POSITION
00274'024710          LDA 1   .177    ; GET MASK
00275'107405          AND     0,1 SNR ; IN COLUMN ZERO?
00276'001400          JMP     0,3     ; YES, DO NOTHING
00277'100404          NEG     0,0     ; MOVE BACK ...
00300'100000          COM     0,0     ; ... ONE CHARACTER
00301'000713          JMP     MOVES+1 ; RESET CURSOR

00302'020471  THT:    LDA 0   .11     ; .11 = HOR. TAB
00303'106414          SEQ     0,1     ; TEST FOR HOR. TAB
00304'000414          JMP     TLF     ; NOT, TEST FOR LINE FEED
00305'021005$         LDA 0   SPOT,2  ; GET POSITION
00306'024676          LDA 1   .177    ; FETCH MASK
00307'107403          AND     0,1     ; GET POSITION IN LINE
00310'021001$         LDA 0   LINE,2  ; COLUMN LIMIT
00311'122112          SLT     1,0     ; PAST END OF LINE?
00312'001400          JMP     0,3     ; YES, RETURN
00313'021005$         LDA 0   SPOT,2  ; GET POSITION
00314'024454          LDA 1   HTX     ; FETCH MASK
00315'123400          AND     1,0     ; MOVE TO TAB STOP
00316'122400          SUB     1,0     ; ADVANCE ONE TAB STOP
00317'000675          JMP     MOVES+1 ; RESET CURSOR

00320'020454  TLF:    LDA 0   .12     ; .12 = LINE FEED
00321'106414          SEQ     0,1     ; TEST FOR LINE FEED
00322'000414          JMP     TVT     ; NOT, TEST FOR VERT. TAB
00323'021005$         LDA 0   SPOT,2  ; GET POSITION
00324'101300          MOVS    0,0     ; LINE NUMBER TO BOTTOM BYTE
00325'101700          INCS    0,0     ; BUMP LINE AND SWAP BACK
00326'041075          STA 0   TEMP,2  ; SAVE POSITION
00327'024645          LDA 1   HMASK   ; GET MASK
00330'107700          ANDS    0,1     ; GET LINE NUMBER
00331'021002$         LDA 0   PAGE,2  ; LINE LIMIT
00332'122112          SLT     1,0     ; PAST END OF PAGE?
00333'001400          JMP     0,3     ; YES, RETURN
00334'021075          LDA 0   TEMP,2  ; GET POSITION
00335'000657          JMP     MOVES+1 ; RESET CURSOR

00336'020650  TVT:    LDA 0   .13     ; .13 = VERT. TAB
00337'106414          SEQ     0,1     ; TEST FOR VERT. TAB
00340'000416          JMP     TFF     ; NOT, TEST FOR FORM FEED
00341'021005$         LDA 0   SPOT,2  ; GET POSITION
00342'024427          LDA 1   VTX     ; FETCH MASK
00343'101300          MOVS    0,0     ; LINE NUMBER TO BOTTOM BYTE
00344'123400          AND     1,0     ; MOVE TO TAB STOP
00345'122700          SUBS    1,0     ; ADVANCE ONE TAB STOP
00346'041075          STA 0   TEMP,2  ; SAVE POSITION
00347'024625          LDA 1   HMASK   ; GET MASK
00350'107700          ANDS    0,1     ; GET LINE NUMBER
```

```
00351'021002$          LDA 0   PAGE,2  ; LINE LIMIT
00352'122112           SLT     1,0     ; PAST END OF PAGE?
00353'031400           JMP     0,3     ; YES, RETURN
00354'021075           LDA 0   TEMP,2  ; GET POSITION
00355'000637  RSET:    JMP     MOVES+1 ; RESET CURSOR

00356'020623  TFF:     LDA 0   .14     ; .14 = FORM FEED
00357'122405           SUB     1,0 SNR ; TEST FOR FORM FEED
00360'000775           JMP     RSET    ; HOME CURSOR

00361'020626           LDA 0   .15     ; .15 = RETURN
00362'136414           SEQ     0,1     ; TEST FOR RETURN
00363'031400           JMP     0,3     ; NOT, ILLEGAL CONTROL CODE
00364'020610           LDA 0   HMASK   ; FETCH MASK
00365'025005$          LDA 1   SPOT,2  ; GET POSITION
00366'123400           AND     1,0     ; MOVE TO BEGINNING OF LINE
00367'000766           JMP     RSET    ; RESET CURSOR

00370'177770  HTX:     177770  ; HOR. TAB MASK
00371'177774  VTX:     177774  ; VERT. TAB MASK

00372'000010  .10:     10      ; BACK SPACE
00373'000011  .11:     11      ; HOR. TAB
00374'000012  .12:     12      ; LINE FEED

              .END
```

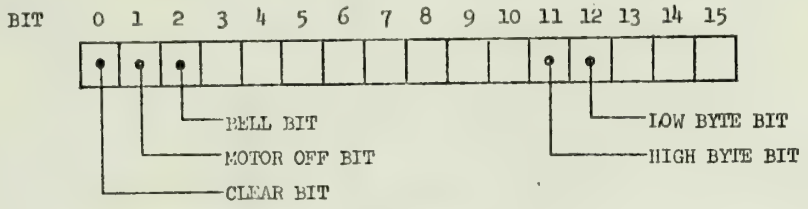

SYMBOL	VALUE	DEF'N	REFERENCES						
AA	000262'	4:57	4:25						
ALL	000130'	3:12	3:08						
BITS	000072	1:14	1:25	1:42	2:11	2:44	3:12	3:27	
BOK	000110'	2:52	2:48						
BON	000065'	2:30	2:14						
BOTH	000123'	3:05	3:01						
CNTRL	000265'	5:03	3:35						
DATA	000073	1:15	1:24	2:56	3:06	3:28	4:12		
DING	000101'	2:44	2:20	2:28	2:34				
DZAP	000000'	1:23	1:20						
EE	000264'	5:01	4:36						
FINI	000144'	3:27	3:21						
HIGH	000125'	3:07	3:03						
HMASK	000174'	3:53	3:44	5:40	5:57	6:14			
HOME	$X	1:10	1:39	3:17					
HTX	000370'	6:19	5:28						
LINE	$X	1:05	3:40	5:24					
LOW	000124'	3:06	3:04						
MASK	000175'	3:54	2:25	2:39					
MODE	$X	1:08	1:40	1:46	1:52	1:55	1:58	2:06	2:09
			2:17	2:24	2:27	2:31	2:38	2:42	
MOVE	000127'	3:10	2:55						
MOVER	000074	1:16	4:22	4:33	4:55				
MOVES	000213'	4:12	3:10	5:16	5:31	5:46	6:05		
OK	000017'	1:42	1:28						
PAR	$X	1:06	3:46	5:42	6:01				
REPO	000224'	4:22	3:25						
RSET	000355'	6:05	6:09	6:17					
RZAP	000056	1:12	1:23	3:29					
SAVE	000077	1:18	3:36	3:49					
SCOL	000237'	4:35	4:20						
SEND	000173'	3:51	1:31	1:33	1:35	1:51	2:05	2:23	2:37
			2:50	3:24	4:24	4:28	4:37	4:39	4:52
			4:54	5:05					
SPOT	$X	1:09	1:38	3:16	3:38	3:43	3:50	4:13	4:14
			4:26	4:30	5:10	5:21	5:27	5:36	5:51
			6:15						
TEMP	000075	1:17	2:16	2:30	2:52	4:35	4:40	5:39	5:45
			5:56	6:04					
TFF	000356'	6:07	5:50						
THT	000302'	5:18	5:09						
TLF	000320'	5:33	5:20						
TOK	000045'	2:11	1:48	1:56	2:02				
TON	000034'	1:58	1:44						
TVT	000336'	5:48	5:35						
TYPE	000147'	3:31	3:05	3:07					
VTX	000371'	6:20	5:52						
ZOP	$X	1:07	3:51						
ZZ	000263'	4:58	4:50						
.10	000372'	6:22	5:07						
.11	000201'	4:01	3:13						
.12	000205'	4:05	4:43						
.11	000373'	6:23	5:18						
.12	000374'	6:24	5:33						
.13	000206'	4:06	4:23	5:48					

/BOL	VALUE	DEF'N	REFERENCES					
13	000176'	3:56	1:30	6:07				
15	000207'	4:07	3:23	6:11				
16	000210'	4:08	2:04					
17	000200'	3:58	1:34	1:50				
177	000244'	4:04	3:31	3:37	4:17	4:31	5:11	5:22
14	000211'	4:09	2:36					
30	000177'	3:57	1:32					
17	000202'	4:02	3:33					
30	000212'	4:10	2:22					
7	000203'	4:03	2:49	2:53	5:03			

FLAGGED LINES: NONE

TERMINET PRINTER OUTPUT (PZAP)

The Terminet printer output routine is used to output to a General Electric Terminet printer. It receives its input characters in AC1 and a flag word in AC0. The action taken due to the flag word is as follows:



If the clear bit is high, a form feed and carriage return are sent.

If the motor off bit is high, the motor is turned off and no data characters are sent. The motor state is normally checked and if off, it is turned on before data is sent.

If the bell bit is high, the routine sends a bell code.

If the high byte bit is high, the routine sends the character in the upper byte of the data word (AC1).

If the low byte bit is high, the routine sends the character in the lower byte of the data word.

If both bits are high, both bytes are sent in the above order.

If neither bit is high, the input data word is taken as a position. If the position is on the current line or the next line, the physical printing position is moved to the position so indicated. If the input position is not on the current line or the next line, two lines are skipped and the device is moved to the proper column. The input position word contains the line number in the upper byte and the column number in the lower byte.

```
PAGE  1          A I D S     -       TERMINET PRINTER OUTPUT
02/20/74
                        .HEAD   A I D S     -       TERMINET PRINTER OUTPUT

                        .TITLE  PZAP

                        .EXTD   PAGE    ; PAGE SIZE
                        .EXTD   LINE    ; LINE LENGTH
                        .EXTD   PADS    ; PADS COUNTER
                        .EXTD   FLAG    ; DEVICE FLAG WORD
                        .EXTD   ZOP     ; OUTPUT DRIVER
                        .EXTD   SPOT    ; CURSOR POSITION
                        .EXTD   HOME    ; OFFICIAL POSITION
                        .EXTD   MODE    ; DISPLAY MODE WORD

              000056    RZAP=   R6      ; RETURN WORD

              000072    BITS=   T2      ; OPTION BITS
              000073    DATA=   T3      ; DATA / POSITION
              000074    MOVER=  T4      ; SUBROUTINE RETURN ADDRESS
              000075    TEMP=   T5      ; TEMPORARY STORAGE
              000076    TRTN=   T6      ; SECONDARY SUBROUTINE RETURN ADDRESS
              000077    SAVE=   T7      ; TEMPORARY SAVE WORD

                        .ENT    PZAP
                        .NREL

00000'055056  PZAP:     STA 3   RZAP,2  ; SAVE RETURN
00001'045073            STA 1   DATA,2  ; SAVE DATA
00002'041072            STA 0   BITS,2  ; AND FLAGS

00003'025010$           LDA 1   MODE,2  ; GET MODE WORD
00004'125122            MOVZL   1,1 SZC ; IS MOTOR ON?
00005'000410            JMP     MON     ; YES, GO ON
00006'125240            MOVOR   1,1     ; SET MOTOR BIT ON
00007'045010$           STA 1   MODE,2  ; RESET MODE WORD

00010'024560            LDA 1   .33     ; GET ESCAPE CODE
00011'007005$           JSR     @ZOP,2  ; SEND IT
00012'024553            LDA 1   ..H     ; GET MOTOR ON COMMAND
00013'007005$           JSR     @ZOP,2  ; SEND IT
00014'004536            JSR     PAD40   ; PAD FOR TIMING

00015'021072  MON:      LDA 0   BITS,2  ; GET FLAG WORD
00016'101103            MOVL    0,0 SNC ; TEST CLEAR BIT
00017'000406            JMP     CLOK    ; IF NOT SET, GO
00020'024551            LDA 1   .15     ; ASCII "RETURN"
00021'007005$           JSR     @ZOP,2  ; MOVE TO COLUMN ZERO
00022'004550            JSR     TOP     ; EJECT A PAGE
00023'126400            SUB     1,1     ; MAKE A ZERO
00024'045006$           STA 1   SPOT,2  ; REPOSITION

00025'021072  CLOK:     LDA 0   BITS,2  ; FETCH FLAG WORD
00026'101120            MOVZL   0,0     ; IGNORE CLEAR BIT
00027'101123            MOVZL   0,0 SNC ; TEST MOTOR OFF FLAG
00030'000414            JMP     MOFF    ; MOTOR IS OK

00031'041076            STA 0   TRTN,2  ; SAVE FLAGS
00032'004520            JSR     PAD40   ; PAD FOR TIMING
00033'024535            LDA 1   .33     ; GET ESCAPE CODE
```

```
                        JSR     @ZOP,2  ; SEND IT
                        LDA 1   ..J     ; GET MOTOR OFF COMMAND
                        JSR     @ZOP,2  ; SEND IT
                        LDA 0   TRTN,2  ; RECOVER FLAGS

                        LDA 1   MODE,2  ; GET MODE WORD
                        MOVZL   1,1     ; REMOVE MOTOR BIT
                        MOVZR   1,1     ; CLEAR MOTOR BIT
                        STA 1   MODE,2  ; RESTORE MODE WORD

MOFF:                   MOVZL   0,0 SNC ; TEST BELL FLAG
                        JMP     OK      ; NO BELL WANTED
                        LDA 1   .7      ; GET A BELL CODE
                        JSR     @ZOP,2  ; SEND IT

OK:                     LDA 0   BITS,2  ; FETCH FLAG WORD
                        MOVZR   0,0     ; IGNORE TAG BIT
                        MOVZR   0,0     ; IGNORE BLINK BIT
                        LDA 1   .7      ; 3-BIT MASK
                        ANDZR   1,0 SNR ; GET MODE BITS
                        JMP     MOVE    ; IF ZERO, GO
                        LDA 1   DATA,2  ; GET DATA
                        MOVS    1,1     ; SWAP BYTES
                        MOVZR#  0,0 SEZ ; BOTH BYTES?
                        JMP     BOTH    ; YES, SEND BOTH
                        MOVZR#  0,0 SZR ; WHICH OF THE TWO?
                        JMP     HIGH    ; SEND THE HIGH BYTE
                        JMP     LOW     ; SEND THE LOW BYTE
BOTH:                   JSR     TYPE    ; SEND A BYTE
LOW:                    LDA 1   DATA,2  ; RECOVER DATA
HIGH:                   JSR     TYPE    ; SEND A BYTE
                        JMP     ALL     ; RETURN

MOVE:                   LDA 0   DATA,2  ; GET NEW POSITION
                        JSR     MOVES   ; REPOSITION CURSOR

ALL:                    LDA 0   BITS,2  ; RESTORE FLAGS
                        LDA 1   .100    ; GET UPDATE MASK
                        AND     0,1 SNR ; UPDATE POSITION?
                        JMP     .+3     ; NO, GO ON
                        LDA 1   SPOT,2  ; GET NEW POSITION
                        STA 1   HOME,2  ; UPDATE POSITION
                        LDA 1   DATA,2  ; RESTORE DATA/POSITION
                        JMP     @RZAP,2 ; RETURN

MOVES:                  LDA 1   SPOT,2  ; GET OLD POSITION
                        SUB#    0,1 SNR ; IF EQUAL,
                        JMP     0,3     ; DON'T BOTHER

                        STA 3   MOVER,2 ; SAVE RETURN
                        STA 0   SAVE,2  ; AND NEW POSITION
                        LDA 3   .377    ; FETCH LOW BYTE MASK
                        AND     3,0     ; GET NEW COLUMN NUMBER
                        AND     3,1     ; AND OLD COLUMN NUMBER
                        SUBZ    1,0 SZC ; COMPUTE NUMBER OF SPACES TO MOVE
                        JMP     MOVE1   ; NORMAL FORWARD MOVE

                        LDA 1   .15     ; CARRIAGE RETURN
```



```
                        JSR     @ZOP,2  ; MOVE TO BEGINNING OF LINE
                        LDA 0   SAVE,2  ; RECOVER NEW POSITION
                        LDA 3   .377    ; FETCH LOW BYTE MASK
MOVE1:                  AND     3,0 SNR ; ANY SPACES TO MOVE?
                        JMP     COK     ; NO, CONTINUE

                        STA 0   TEMP,2  ; SAVE SPACE COUNTER
                        LDA 1   .40     ; ASCII "SPACE"
                        JSR     @ZOP,2  ; MOVE WITH A SPACE
                        DSZ     TEMP,2  ; DONE MOVING?
                        JMP     .-3     ; NO, MOVE AGAIN

COK:                    LDA 0   SAVE,2  ; GET NEW POSITION
                        LDA 1   SPOT,2  ; AND OLD POSITION
                        STA 0   SPOT,2  ; UPDATE OLD POSITION
                        LDA 3   H377    ; GET UPPER BYTE MASK
                        ANDS    3,0     ; GET NEW LINE NUMBER
                        ANDS    3,1     ; AND OLD LINE NUMBER
                        SUB     1,0 SNR ; COMPUTE NUMBER OF LINES TO MOVE
                        JMP     @MOVER,2; ZERO, DO NOTHING
                        MOVZR   0,0 SZR ; JUST ONE, OR TWO?
                        JSR     LYNEF   ; FEED TWICE IF NOT
                        JSR     LYNEF   ; OTHERWISE ONLY ONCE
                        JMP     @MOVER,2; THEN RETURN

PAD:                    LDA 0   PADS,2  ; LINE/CHAR COUNTER
                        LDA 1   H377    ; FETCH MASK
                        AND     0,1     ; EXTRACT LINES
                        STA 1   PADS,2  ; AND RESTORE
                        LDA 1   .377    ; FETCH MASK
                        AND     1,0 SKP ; EXTRACT CHARACTERS

PAD40:                  SUB     0,0     ; SET TO ZERO FOR FULL PADDING
                        LDA 1   .40     ; GET MAXIMUM PADDING
                        SUBZ    1,0 SZC ; COMPUTE NUMBER NEEDED
                        JMP     0,3     ; NONE NEEDED, RETURN

PADX:                   STA 3   SAVE,2  ; SAVE RETURN
                        STA 0   TEMP,2  ; AND COUNTER
                        SUB     1,1     ; CREATE A ZERO
                        JSR     @ZOP,2  ; PAD WITH A NULL
                        ISZ     TEMP,2  ; DONE PADDING?
                        JMP     .-3     ; NO, DO IT AGAIN
                        JMP     @SAVE,2 ; YES, RETURN

..H:                    "H              ; MOTOR ON COMMAND
..J:                    "J              ; MOTOR OFF COMMAND
.40:                    40              ; ASCII SPACE (STOP BLINK)
.33:                    33              ; ESCAPE CODE
.15:                    15              ; ASCII "RETURN"

TOP:                    JMP     FORM    ; LINK TO FORM

TYPE:                   LDA 0   .177    ; FETCH MASK
                        AND     0,1     ; EXTRACT DATA
                        LDA 0   .37     ; HIGHEST CONTROL CODE
                        SGT     1,0     ; TEST FOR CONTROL CHARACTER
                        JMP     CNTRL   ; IS CONTROL
```

```
                        STA I   SAVE,2  ; SAVE DATA WORD
                        LDA 0   .177    ; FETCH MASK
                        LDA I   SPOT,2  ; GET POSITION
                        AND     0,1     ; GET POSITION IN LINE
                        LDA 0   LINE,2  ; COLUMN LIMIT
                        SLT     1,0     ; PAST END OF LINE?
                        JMP     0,3     ; YES, RETURN
                        LDA 0   PADS,2  ; LINE/CHAR COUNTER
                        LDA I   H377    ; GET HI-BYTE MASK
                        ANDS    0,1     ; EXTRACT LINE NUMBER
                        LDA 0   PAGE,2  ; PAGE SIZE
                        SLT     1,0     ; PAST END OF PAGE?
                        JMP     0,3     ; YES, RETURN
                        LDA I   SAVE,2  ; RESTORE DATA WORD
                        ISZ     PADS,2  ; BUMP COUNTER
                        ISZ     SPOT,2  ; BUMP POSITION
                        JMP     @ZOP,2  ; SEND THE DATA

                .7:     7               ; 3-BIT MASK
                .37:    37              ; US = FORMAT OFF
                .10M:   100             ; BIT 9 IS UPDATE FLAG
                .177:   177             ; 7-BIT MASK
                .377:   377             ; LOWER BYTE MASK
                H377:   377*400         ; UPPER BYTE MASK

                CHTRL:  LDA 0   .7      ; .7 = BELL
                        SNE     0,1     ; TEST FOR BELL
                        JMP     @ZOP,2  ; IS A BELL, SEND IT

                        LDA 0   .10     ; .10 = BACK SPACE
                        SEQ     0,1     ; TEST FOR BACK SPACE
                        JMP     THT     ; NOT, TEST FOR HOR. TAB
                        LDA 0   SPOT,2  ; PICK UP POSITION
                        LDA I   .177    ; GET MASK
                        AND     0,1 SNR ; IN COLUMN ZERO?
                        JMP     0,3     ; YES, DO NOTHING
                        DSZ     SPOT,2  ; DECREMENT POSITION
                        LDA 0   SPOT,2  ; GET POSITION
                        JMP     MOVES   ; RESET CURSOR

                THT:    LDA 0   .11     ; .11 = HOR. TAB
                        SEQ     0,1     ; TEST FOR HOR. TAB
                        JMP     TLF     ; NOT, TEST FOR LINE FEED
                        LDA 0   SPOT,2  ; GET POSITION
                        LDA I   .177    ; FETCH MASK
                        AND     0,1     ; GET POSITION IN LINE
                        LDA 0   LINE,2  ; COLUMN LIMIT
                        SGT     0,1     ; PAST END OF LINE?
                        JMP     0,3     ; YES, RETURN
                        LDA 0   SPOT,2  ; GET POSITION
                        LDA I   HTX     ; FETCH MASK
                        AND     1,0     ; MOVE TO TAB STOP
                        SUB     1,0     ; ADVANCE ONE TAB STOP
                        JMP     MOVES   ; RESET CURSOR

                TLF:    LDA 0   .12     ; .12 = LINE FEED
                        SEQ     0,1     ; TEST FOR LINE FEED
                        JMP     TVT     ; NOT, TEST FOR VERT. TAB
```



```
                LINEF:  LDA 0   SPOT,2  ; GET POSITION
                        MOVS    0,0     ; LINES TO BOTTOM BYTE
                        INCS    0,0     ; BUMP AND RESTORE LINES
                        STA 0   SPOT,2  ; NEW POSITION

                LYNEF:  LDA 0   PADS,2  ; LINE/CHAR COUNTER
                        LDA I   H377    ; FETCH MASK
                        SUB     1,0     ; INCREMENT LINE
                        STA 0   PADS,2  ; AND RESTORE
                        ANDS    1,0     ; EXTRACT LINES
                        LDA I   PAGE,2  ; GET PAGE SIZE
                        SLT     0,1     ; ROOM ON THIS PAGE?
                        JMP     FORM    ; NO, ISSUE FORM FEED

                        STA 3   TRTN,2  ; SAVE RETURN
                        JSR     PAD     ; PAD FOR TIMING
                        LDA 3   TRTN,2  ; RETURN ADDRESS
                        LDA I   .12     ; ASCII LINE FEED
                        JMP     @ZOP,2  ; MOVE TO NEXT LINE

                TVT:    LDA 0   .13     ; .13 = VERT. TAB
                        SEQ     0,1     ; TEST FOR VERT. TAB
                        JMP     TFF     ; NOT, TEST FOR FORM FEED

                        STA 3   SAVE,2  ; SAVE RETURN
                VTL:    JSR     LINEF   ; ISSUE LINE FEED
                        LDA 0   SPOT,2  ; GET POSITION
                        LDA I   VTX     ; AND VERT. TAB MASK
                        AND     1,0 SZR ; THIS LINE A TAB STOP?
                        JMP     VTL     ; NO, TRY NEXT LINE
                        JMP     @SAVE,2 ; YES, RETURN

                TFF:    LDA 0   .14     ; .14 = FORM FEED
                        SEQ     0,1     ; TEST FOR FORM FEED
                        JMP     TCR     ; NOT, TEST FOR RETURN

                FORM:   STA 3   TRTN,2  ; SAVE RETURN
                        JSR     PAD     ; PAD FOR TIMING
                        LDA I   .13     ; GET ASCII "VT"
                        LDA 0   FLAG,2  ; DEVICE FLAG WORD
                        MOVZR   0,0 SNC ; FULL PAGE EJECT?
                        INC     1,1     ; YES, CHANGE "VT" TO "FF"
                        JSR     @ZOP,2  ; AND SEND IT

                        LDA 0   PADS,2  ; LINE/CHAR COUNTER
                        LDA I   H377    ; FETCH MASK
                        ANDS    1,0     ; EXTRACT LINES
                        LDA I   PAGE,2  ; GET PAGE SIZE
                        SGT     0,1     ; IF LOST, ASSUME FULL
                        SUB     0,1     ; COMPUTE LINES TO END OF PAGE
                        NEGOL   1,0     ; COMPUTE NEGATIVE PAD COUNT
                        SUB     1,0 SZR ; IF NOT ZERO,
                        JSR     PADX    ; PAD 3 FOR EACH LINE LEFT

                        LDA I   SPOT,2  ; GET POSITION
                        LDA 0   .377    ; FETCH MASK
                        AND     0,1     ; SET TO LINE ZERO
```

```
00346'045006$          STA 1   SPOT,2  ; AND RESTORE
00347'126400           SUB     1,1     ; CREATE A ZERO
00350'045003$          STA 1   PADS,2  ; CLEAR LINE/CHAR COUNTER
00351'003076           JMP     @TRTN,2 ; AND RETURN

00352'020617  TCR:     LDA Ø   .15     ; .15 = RETURN
00353'106414           SEQ     Ø,1     ; TEST FOR RETURN
00354'001400           JMP     Ø,3     ; NOT, ILLEGAL CONTROL CODE

00355'020651           LDA Ø   H377    ; FETCH MASK
00356'025006$          LDA 1   SPOT,2  ; GET POSITION
00357'123400           AND     1,Ø     ; SET TO COLUMN ZERO
00360'041006$          STA Ø   SPOT,2  ; AND RESTORE
00361'024610           LDA 1   .15     ; ASCII RETURN
00362'003005$          JMP     @ZOP,2  ; MOVE TO COLUMN ZERO

00363'000010  .1Ø:     1Ø              ; BACK SPACE
00364'000011  .11:     11              ; HOR. TAB
00365'000012  .12:     12              ; LINE FEED
00366'000013  .13:     13              ; VT = LINE ADDRESS
00367'000014  .14:     14              ; FF = SCREEN ERASE

00370'177770  HTX:     -1Ø     ; HOR. TAB MASK
00371'001400  VTX:     3*400   ; VERT. TAB MASK

              .END
```


SYMBOL	VALUE	DEF'N	REFERENCES						
ALL	000073'	2:37	2:32						
BITS	000072	1:16	1:28	1:42	1:51	2:16	2:37		
BOTH	000065'	2:29	2:25						
CLOK	000025'	1:51	1:44						
CNTRL	000227'	4:26	3:58						
COK	000130'	3:13	3:05						
DATA	000073	1:17	1:27	2:22	2:30	2:34	2:43		
FLAG	$X	1:08	5:41						
FORM	000323'	5:38	3:52	5:14					
H377	000226'	4:24	3:16	3:27	4:09	5:08	5:47	6:10	
HIGH	000067'	2:31	2:27						
HOME	$X	1:11	2:42						
HTX	000370'	6:23	4:51						
LINE	$X	1:06	4:05	4:47					
LINEF	000265'	5:02	5:27						
LOW	000066'	2:30	2:28						
LYNEF	000271'	5:07	3:22	3:23					
MODE	$X	1:12	1:30	1:34	2:06	2:09			
MOFF	000044'	2:11	1:54						
MON	000015'	1:42	1:32						
MOVE	000071'	2:34	2:21						
MOVEI	000121'	3:04	2:56						
MOVER	000074	1:18	2:50	3:20	3:24				
MOVES	000103'	2:46	2:35	4:39	4:54				
OK	000050'	2:16	2:12						
PA	000144'	3:26	5:17	5:39					
PAD40	000152'	3:33	1:40	1:57					
PADS	$X	1:07	3:26	3:29	4:08	4:15	5:07	5:10	5:46
			6:03						
PADX	000156'	3:38	5:54						
PAGE	$X	1:05	4:11	5:12	5:49				
PZAP	000000'	1:26	1:23						
RZAP	000056	1:14	1:26	2:44					
SAVE	000077	1:21	2:51	3:02	3:13	3:38	3:44	4:01	4:14
			5:26	5:32					
SPOT	$X	1:10	1:49	2:41	2:46	3:14	3:15	4:03	4:16
			4:33	4:37	4:38	4:44	4:50	5:02	5:05
			5:28	5:56	6:01	6:11	6:13		
TCR	000352'	6:06	5:36						
TEMP	000075	1:19	3:07	3:10	3:39	3:42			
TFF	000320'	5:34	5:24						
THT	000244'	4:41	4:32						
TLF	000262'	4:56	4:43						
TOP	000172'	3:52	1:47						
TRTN	000076	1:20	1:56	2:04	5:16	5:18	5:38	6:04	
TVT	000306'	5:22	4:58						
TYPE	000173'	3:54	2:29	2:31					
VTL	000312'	5:27	5:31						
VTX	000371'	6:24	5:29						
ZOP	$X	1:09	1:37	1:39	1:46	2:01	2:03	2:14	3:01
			3:09	3:41	4:17	4:28	5:20	5:44	6:15
.1Ø	000363'	6:17	4:30						
.100	000223'	4:21	2:38						
.11	000364'	6:18	4:41						
.12	000365'	6:19	4:56	5:19					

SYMBOL	VALUE	DEF'N	REFERENCES			
I3	000366'	6:20	5:22	5:40		
I4	000367'	6:21	5:34			
I5	000171'	3:50	1:45	2:58	6:06	6:14
I77	000224'	4:22	3:54	4:02	4:34	4:45
I8	000173'	3:49	1:36	1:58		
I7	000222'	4:20	3:56			
I77	000225'	4:23	2:52	3:03	3:30	5:57
I0	000167'	3:48	3:08	3:34		
I	000221'	4:19	2:13	2:19	4:26	
I9	000165'	3:46	1:38			
I3	000166'	3:47	2:02			

FLAGGED LINES: NONE

SYMBOL	VALUE	DEF'N	REFERENCES	
..	[illegible]	1:13	1:10	
.1	[illegible]	1:17	1:21	
.2	[illegible]	1:26	1:16	
.3	[illegible]	1:27	1:13	1:24

FLAGGED LINES: NONE

DIVIDE SUBROUTINE (DIV.)

The divide subroutine performs an unsigned divide as follows:

INPUT: AC0: divisor
AC1: dividend

OUTPUT: AC0: remainder
AC1: quotient
AC2: unchanged
CARRY: 0 if OK, 1 if error

ERROR: divide by zero sets carry high


```
                        .HEAD   A I D S      -       DIVIDE SUBROUTINE
                        .TITLE  DIV.
                        ;       IN: AC0 - DIVISOR       OUT: AC0 - REMAINDER
                        ;           AC1 - DIVIDEND           AC1 - QUOTIENT
                        ;                                    AC2 - RESTORED
                        ;       ERROR:  DIVIDE BY ZERO       CARRY -  0 IF OK
                        ;               SETS CARRY HIGH                1 IF ERROR
                        .ENT    DIV.
                        .NREL
00000'054421    DIV.:   STA 3   DIV.4   ; SAVE RETURN ADDRESS
00001'111005            MOV     0,2 SNR ; DIVISOR TO AC2
00002'000413            JMP     DIV.2   ; ERROR, DIVIDE BY ZERO
00003'102400            SUB     0,0     ; CLEAR AC0
00004'034414            LDA 3   DIV.3   ; 16 ITERATIONS
00005'125120            MOVZL   1,1     ; SHIFT LOW DIVIDEND
00006'101100    DIV.1:  MOVL    0,0     ; SHIFT HIGH DIVIDEND
00007'142412            SUB#    2,0 SZC ; DOES DIVISOR GO IN?
00010'142400            SUB     2,0     ; YES
00011'125100            MOVL    1,1     ; SHIFT LOW DIVIDEND
00012'175404            INC     3,3 SZR ; CHECK COUNT
00013'000773            JMP     DIV.1   ; NOT DONE
00014'176441            SUBO    3,3 SKP ; DONE, CLEAR CARRY
00015'176420    DIV.2:  SUBZ    3,3     ; ERROR, SET CARRY
00016'030002            LDA 2   2       ; RESTORE AC2
00017'002402            JMP     @DIV.4  ; RETURN

00020'177760    DIV.3:  -16.            ; ITERATION COUNT
00021'000000    DIV.4:  0               ; RETURN ADDRESS

                        .END
```

SYMBOL VALUE DEF'N REFERENCES

..	000003'	1:15	1:12	
V.1	000006'	1:21	1:26	
..2	000015'	1:28	1:17	
..3	000020'	1:32	1:19	
V.4	000021'	1:33	1:15	1:30

FLAGGED LINES: NONE

ASCII CONVERSION (CVA.)

The ASCII conversion routine takes a binary value in AC1 and converts this numeric value into five decimal digits which it stores in the parameter words of the terminal information table for use by the calling program.

INPUT: AC1: binary number

OUTPUT: P7: ASCII units character
P6: ASCII tens character
P5: ASCII hundreds character
P4: ASCII thousands character
P3: ASCII ten-thousands character


```
                        .HEAD   A I D S     A S C I I       C O N V E R S I O N

                        .TITLE  CVA.

                        .EXTD   .DIV    ; DIVIDE ROUTINE ENTRY

                        .ENT    CVA.
                        .NREL

000000'054424   CVA.:   STA 3   CVT.4    ; SAVE RETURN ADDRESS
000001'004413           JSR     CVT.1    ; GET UNITS
000002'041047           STA 0   P7,2     ; STORE UNITS
000003'004411           JSR     CVT.1    ; GET TENS
000004'041046           STA 0   P6,2     ; STORE TENS
000005'004407           JSR     CVT.1    ; GET HUNDREDS
000006'041045           STA 0   P5,2     ; STORE HUNDREDS
000007'004405           JSR     CVT.1    ; GET THOUSANDS
000010'041044           STA 0   P4,2     ; STORE THOUSANDS
000011'004403           JSR     CVT.1    ; GET TEN-THOUSANDS
000012'041043           STA 0   P3,2     ; STORE TEN-THOUSANDS
000013'002411           JMP     @CVT.4   ; RETURN

000014'054411   CVT.1:  STA 3   CVT.5    ; SAVE RETURN ADDRESS
000015'020405           LDA 0   CVT.2    ; GET DIVISOR (10.)
000016'006001$          JSR     @.DIV    ; DIVIDE
000017'034404           LDA 3   CVT.3    ; ASCII "ZERO"
000020'163000           ADD     3,0      ; CONVERT TO ASCII DIGIT
000021'002404           JMP     @CVT.5   ; RETURN

000022'000012   CVT.2:  10.      ; DECIMAL 10 (DIVISOR)
000023'000060   CVT.3:  "0       ; ASCII "ZERO" (ADDEND)
000024'000000   CVT.4:  0        ; RETURN ADDRESS
000025'000000   CVT.5:  0        ; SECONDARY RETURN

                        .END
```

SYMBOL	VALUE	DEF'N	REFERENCES				
.0	000003'	1:10	1:07				
.1	000014'	1:23	1:11	1:13	1:15	1:17	1:19
.2	000022'	1:30	1:24				
.3	000023'	1:31	1:26				
.4	000024'	1:32	1:10	1:21			
.5	000025'	1:33	1:23	1:28			
.IV	$X	1:05	1:25				

FLAGGED LINES: NONE

BINARY CONVERSION (CVB.)

The binary conversion routine takes five ASCII characters out of the parameter words of the terminal information table and extracts from them a binary number which is returned in AC1. The routine will ignore leading non-digit characters but will terminate conversion if a non-leading non-digit is encountered.

INPUT: P3: highest order ASCII character
P4:
P5: intermediate ASCII characters
P6:
P7: lowest order ASCII character

OUTPUT: AC1: binary value


```
                        .HEAD   A I D S     B I N A R Y     C O N V E R S I O N
                        .TITLE  CVB.
                        .ENT    CVB.
                        .NREL
00000'054420  CVB.:     STA 3   RTRN    ; SAVE RETURN ADDRESS
00001'102400            SUB     0,0     ; CREATE A ZERO
00002'040421            STA 0   FLAG    ; CLEAR LEADING ZERO FLAG
00003'040417            STA 0   VALUE   ; CLEAR RESULT

00004'025043            LDA 1   P3,2    ; GET 1ST ASCII CHARACTER
00005'004417            JSR     TASK    ; PROCESS IT

00006'025044            LDA 1   P4,2    ; GET 2ND ASCII CHARACTER
00007'004415            JSR     TASK    ; PROCESS IT

00010'025045            LDA 1   P5,2    ; GET 3RD ASCII CHARACTER
00011'004413            JSR     TASK    ; PROCESS IT

00012'025046            LDA 1   P6,2    ; GET 4TH ASCII CHARACTER
00013'004411            JSR     TASK    ; PROCESS IT

00014'025047            LDA 1   P7,2    ; GET 5TH ASCII CHARACTER
00015'004407            JSR     TASK    ; PROCESS IT

00016'024404  DONE:     LDA 1   VALUE   ; GET BINARY VALUE
00017'002401            JMP     @RTRN   ; RETURN

00020'000000  RTRN:     0       ; RETURN ADDRESS
00021'000000  SRET:     0       ; SUBROUTINE RETURN ADDRESS

00022'000000  VALUE:    0       ; BINARY CONVERSION RESULT
00023'000000  FLAG:     0       ; LEADING ZERO FLAG

00024'054775  TASK:     STA 3   SRET    ; SAVE RETURN ADDRESS
00025'020424            LDA 0   .177    ; FETCH MASK
00026'107400            AND     0,1     ; EXTRACT LOW BYTE
00027'020423            LDA 0   .60     ; ASCII "0"
00030'034423            LDA 3   .71     ; ASCII "9"
00031'106513            SGT     0,1     ; LESS THAN A DIGIT?
00032'136512            SLE     1,3     ; MORE THAN A DIGIT?
00033'000412            JMP     NON     ; NOT A DIGIT

00034'010767            ISZ     FLAG    ; BUMP FLAGS
00035'106400            SUB     0,1     ; CONVERT TO BINARY
00036'020764            LDA 0   VALUE   ; GET PARTIAL RESULT

00037'115120            MOVZL   0,3     ; MULTIPLY PARTIAL ...
00040'175120            MOVZL   3,3     ; ... RESULT ...
00041'163120            ADDZL   3,0     ; ... BY TEN

00042'123000            ADD     1,0     ; COMPUTE NEW PARTIAL RESULT
00043'040757            STA 0   VALUE   ; STORE IT
00044'002755            JMP     @SRET   ; RETURN

00045'020756  NON:      LDA 0   FLAG    ; GET FLAG WORD
```

```
000016'131004          MOV     0,0 SZR ; LEADING NON-DIGIT?
000047'000747          JMP     DONE    ; NO, TRAILING - EXIT
000050'002751          JMP     @SRET   ; YES, JUST IGNORE IT

000051'000177  .177:   177     ; LOW BYTE MASK
000052'000060  .60:    60      ; ASCII "0"
000053'000071  .71:    71      ; ASCII "9"

               .END
```


SYMBOL	VALUE	DEF'N	REFERENCES				
CVB.	000000'	1:08	1:05				
DONE	000016'	1:28	2:02				
FLAG	000023'	1:35	1:10	1:46	1:58		
NON	000045'	1:58	1:44				
RTRN	000020'	1:31	1:08	1:29			
SRET	000021'	1:32	1:37	1:56	2:03		
TASK	000024'	1:37	1:14	1:17	1:20	1:23	1:26
VALUE	000022'	1:34	1:11	1:28	1:48	1:55	
.177	000051'	2:05	1:38				
.60	000052'	2:06	1:40				
.71	000053'	2:07	1:41				

FLAGGED LINES: NONE

```
                .HEAD   A I D S     C O N T R O L   C E N T E R   L O O K U P

                .TITLE  CCN.

                .EXTD   CCTB,CCTE       ; CONTROL CENTER TABLE LIMITS

                .ENT    CCN.
                .NREL

000000'054426   CCN.:   STA 3   ENTRY    ; SAVE RETURN
000001'024423           LDA 1   MASK    ; HI-BYTE MASK
000002'107403           AND     0,1     ; EXTRACT
000003'044422           STA 1   CCN     ; CENTER NAME

000004'034001$          LDA 3   CCTB    ; CONTROL TABLE BEGIN
000005'022002$ LOOP:    LDA 0   CCTE    ; CONTROL TABLE END
000006'116513           SGT     0,3     ; IF PAST END,
000007'000413           JMP     ERR     ; DIRECT RETURN

000010'021400           LDA 0   0,3     ; CENTER NAME
000011'024413           LDA 1   MASK    ; HI-BYTE MASK
000012'107403           AND     0,1     ; EXTRACT
000013'020412           LDA 0   CCN     ; DESIRED NAME
000014'106415           SNE     0,1     ; IF EQUAL,
000015'000404           JMP     OUT     ; SUCCESS!

000016'020405           LDA 0   SIZE    ; TABLE ENTRY SIZE
000017'117000           ADD     0,3     ; MOVE TO NEXT ENTRY
000020'000765           JMP     LOOP    ; REPEAT
000021'034405   OUT:    ISZ     ENTRY   ; BUMP ADDRESS
000022'000424   ERR:    JMP     @ENTRY  ; RETURN

000023'000040   SIZE:   40      ; ENTRY LENGTH
000024'177400   MASK:   377*400 ; HI-BYTE MASK
000025'000000   CCN:    0       ; DESIRED CENTER
000026'000000   ENTRY:  0       ; RETURN ADDRESS

                .END
```


SYMBOL	VALUE	DEF'N	REFERENCES		
CCN	000025'	1:35	1:13	1:23	
CCN.	000000'	1:10	1:07		
CCTB	$X	1:05	1:15		
CCTE	$X	1:05	1:16		
ENTRY	000026'	1:36	1:10	1:30	1:31
ERR	000022'	1:31	1:18		
LOOP	000005'	1:16	1:29		
MASK	000024'	1:34	1:11	1:21	
OUT	000021'	1:30	1:25		
SIZE	000023'	1:33	1:27		

FLAGGED LINES: NONE

UNIT STATUS ACCESS (USA.)

The Unit Status Access routine is used to locate, delete or insert entries in the unit status table. Each of these functions has its own entry point as follows:

US1.: Insert an entry
US2.: Delete an entry
US3.: Locate an entry

The control center name is input in AC0.

The wanted unit number is input in AC1.

The locate and insert functions return the address of the wanted entry in AC3.


```
                        .HEAD   A I D S    U N I T   S T A T U S   A C C E S S
                        .TITLE  USA.
000040          SIZE=   40      ; STATUS TABLE ENTRY SIZE
                        .EXTD   USTB    ; UNIT STATUS TABLE BEGINNING
                        .EXTD   USTE    ; UNIT STATUS TABLE END
                        .EXTD   TSTB    ; TICKET STATUS TABLE BEGINNING
                        .ENT    US1.    ; UNIT STATUS INSERT
                        .ENT    US2.    ; UNIT STATUS DELETE
                        .ENT    US3.    ; UNIT STATUS LOCATE
                        .NREL
                        ;       UNIT STATUS LOCATE
000000'054413   US3.:   STA 3   RTRN    ; SAVE RETURN ADDRESS
000001'004520           JSR     SERCH   ; LOOK FOR MATCHING ENTRY
000002'000402           SKIP            ; NO MATCH, ERROR RETURN
000003'010410   OUT:     ISZ     RTRN    ; FOUND IT, SUCCESS
000004'020403           LDA 0   NAME    ; RESTORE AC0
000005'024403           LDA 1   NUMB    ; RESTORE AC1
000006'002405           JMP     @RTRN   ; RETURN
000007'000001   NAME:    0       ; CURRENT CONTROL CENTER NAME (CCN)
000010'000000   NUMB:    0       ; WANTED UNIT NUMBER
000011'000000   NEW:     0       ; ADDRESS OF NEW RECORD
000012'000000   COUNT:   0       ; LOOP COUNTER
000013'000000   RTRN:    0       ; RETURN ADDRESS
000014'000000   SRET:    0       ; SUBROUTINE RETURN ADDRESS
                        ;       UNIT STATUS DELETE
000015'054776   US2.:   STA 3   RTRN    ; SAVE RETURN ADDRESS
000016'004503           JSR     SERCH   ; LOOK FOR MATCHING ENTRY
000017'000765           JMP     OUT+1   ; NO MATCH, ERROR RETURN
000020'020500           LDA 0   .SIZE   ; 20 WORD ENTRIES
000021'117000           ADD     0,3     ; POINT TO NEXT ENTRY
000022'020002$  TEST2:  LDA 0   USTE    ; GET END OF TABLE
000023'116513           SGT     0,3     ; MORE TO MOVE?
000024'000411           JMP     MOVED   ; NO, CONTINUE
000025'020473           LDA 0   .SIZE   ; 20 WORD ENTRIES
000026'040764           STA 0   COUNT   ; SET UP COUNTER
000027'025400   NEXT2:  LDA 1   0,3     ; GET TABLE WORD
000030'045740           STA 1   -SIZE,3 ; MOVE IT UP ONE RECORD
000031'175400           INC     3,3     ; MOVE TO NEXT WORD
000032'014760           DSZ     COUNT   ; DONE WITH THIS ENTRY?
000033'000774           JMP     NEXT2   ; NO, DO NEXT WORD
000034'000766           JMP     TEST2   ; YES, TEST FOR END OF TABLE
000035'020002$  MOVED:  LDA 0   USTE    ; GET END OF TABLE
000036'024462           LDA 1   .SIZE   ; 20 WORD ENTRIES
000037'122400           SUB     1,0     ; ALLOW FOR DELETED ENTRY
```

```
                        STA 0   USTE    ; UPDATE END POINTER
                        JMP     OUT     ; TAKE SUCCESS EXIT

                ;       UNIT STATUS INSERT

                USI.:   STA 3   RTRN    ; SAVE RETURN ADDRESS
                        STA 0   NAME    ; SAVE CCN
                        STA 1   NUMB    ; SAVE WANTED UNIT NUMBER
                        LDA 3   USTB    ; GET BEGINNING OF TABLE
                        LDA 0   USTE    ; AND END OF TABLE
                        SGT     0,3     ; NULL TABLE?
                        JMP     PAST1   ; YES, DO IT NOW
                        JSR     SERCH+2 ; LOOK FOR MATCHING ENTRY
                        SKIP            ; NOT PRESENT, CONTINUE
                        JMP     OUT     ; ALREADY PRESENT, RETURN

                PAST1:  STA 3   NEW     ; SAVE TABLE POINTER
                        LDA 3   USTE    ; GET END OF TABLE
                        LDA 0   .SIZE   ; 20 WORD ENTRIES
                        ADD     0,3     ; MOVE END OF TABLE
                        LDA 1   TSTB    ; GET BEGINNING OF TICKETS
                        SLE     3,1     ; WILL NEW ENTRY FIT?
                        JMP     OUT+1   ; NO, TAKE ERROR RETURN
                        STA 3   USTE    ; UPDATE END OF TABLE POINTER

                TEST1:  LDA 0   NEW     ; GET ADDRESS OF NEW ENTRY
                        SGT     3,0     ; HAS SPACE BEEN ALLOWED YET?
                        JMP     FILL    ; YES, INSERT NEW ENTRY
                        LDA 0   .SIZE   ; 20 WORD ENTRIES
                        STA 0   COUNT   ; SET UP COUNTER

                NEXT1:  LDA 1   -SIZE,3 ; GET WORD TO BE MOVED
                        STA 1   0,3     ; MOVE IT UP ONE RECORD
                        INC     3,3     ; POINT TO NEXT WORD
                        DSZ     COUNT   ; DONE WITH THIS ENTRY?
                        JMP     NEXT1   ; NO, MOVE NEXT WORD

                        LDA 0   .SIZE
                        ADD     0,0     ; TWO 20 WORD ENTRIES
                        SUB     0,3     ; POINT BACK TWO ENTRIES
                        JMP     TEST1   ; TEST IF DONE WITH MOVING

                FILL:   LDA 3   NEW     ; SET UP BASE REGISTER
                        LDA 0   .SIZE   ; 20 WORD ENTRIES
                        STA 0   COUNT   ; SET UP COUNTER
                        SUB     0,0     ; CREATE A ZERO
                        STA 0   0,3     ; CLEAR A WORD
                        INC     3,3     ; BUMP ADDRESS
                        DSZ     COUNT   ; KICK COUNTER
                        JMP     .-3     ; AND REPEAT

                        LDA 3   NEW     ; RESTORE ENTER ADDRESS
                        LDA 0   NAME    ; GET CONTROL CENTER NAME
                        STA 0   0,3     ; INSTALL IT
                        LDA 0   NUMB    ; GET UNIT NUMBER
                        STA 0   1,3     ; INSTALL IT
                        JMP     OUT     ; YES, TAKE SUCCESS RETURN
```



```
00120'000040    .SIZE:  SIZE            ; 20 WORDS PER ENTRY


                ;       TABLE SEARCHING ROUTINE

00121'040666    SERCH:  STA 0   NAME    ; SAVE CONTROL CENTER NAME
00122'044666            STA 1   NUMB    ; SAVE DESIRED UNIT NUMBER
00123'054671            STA 3   SRET    ; SAVE RETURN ADDRESS
00124'034001$           LDA 3   USTB    ; SET UP BASE REGISTER

00125'021401    LOOP:   LDA 0   1,3     ; GET UNIT FOR THIS ENTRY
00126'106053            SHS     0,1     ; MORE TO SEARCH?
00127'000404            JMP     NOPE    ; YES, CONTINUE

00130'106415            SNE     0,1     ; IS THIS IT?
00131'010663            ISZ     SRET    ; YES, BUMP RETURN WORD
00132'002662            JMP     @SRET   ; AND RETURN

00133'020765    NOPE:   LDA 0   .SIZE   ; 20 WORD ENTRIES
00134'117000            ADD     0,3     ; POINT TO NEXT ENTRY
00135'020002$           LDA 0   USTE    ; GET END OF TABLE
00136'116512            SLE     0,3     ; MORE TO TRY?
00137'000766            JMP     LOOP    ; YES, DO IT
00140'002654            JMP     @SRET   ; TABLE EXHAUSTED, FAIL RETURN


                .END
```

SYMBOL	VALUE	DEF'N	REFERENCES						
[illegible]	[illegible]12'	1:31	1:47	1:52	2:30	2:35	2:45	2:49	
[illegible]	[illegible]02'	2:43	2:28						
[illegible]	[illegible]25'	3:10	3:22						
[illegible]	[illegible]35'	1:56	1:45						
[illegible]	[illegible]07'	1:28	1:24	2:07	2:53	3:05			
[illegible]	[illegible]11'	1:30	2:17	2:26	2:43	2:52			
[illegible]	[illegible]71'	2:32	2:36						
[illegible]	[illegible]27'	1:49	1:53						
[illegible]	[illegible]33'	3:18	3:12						
[illegible]	[illegible]10'	1:29	1:25	2:08	2:55	3:06			
[illegible]	[illegible]43'	1:23	1:39	2:02	2:15	2:23	2:57		
[illegible]	[illegible]54'	2:17	2:12						
[illegible]	[illegible]13'	1:32	1:19	1:23	1:26	1:37	2:06		
[illegible]	[illegible]21'	3:05	1:20	1:38	2:13				
[illegible]	[illegible]4	1:05	1:50	2:32	3:01				
[illegible]	[illegible]14'	1:33	3:07	3:15	3:16	3:23			
[illegible]	[illegible]64'	2:26	2:41						
[illegible]	[illegible]22'	1:43	1:54						
[illegible]	$X	1:09	2:21						
[illegible]	[illegible]42'	2:06	1:11						
[illegible]	[illegible]15'	1:37	1:12						
[illegible]	[illegible]3'	1:19	1:13						
[illegible]	$X	1:07	2:09	3:08					
[illegible]	$X	1:08	1:43	1:56	2:01	2:10	2:18	2:24	3:20
[illegible]	[illegible]20'	3:01	1:40	1:46	1:57	2:19	2:29	2:38	2:44
			3:18						

FLAGGED LINES: NONE

TICKET STATUS ACCESS (TSA.)

The Ticket Status Access routine is used to locate, delete or insert entries in the ticket status table. Each of these functions has its own entry point as follows:

TS1.: Insert an entry

TS2.: Delete an entry

TS3.: Locate an entry

The control center name is input in AC0.

The wanted ticket number is input in AC1.

The locate and insert functions return the address of the wanted entry in AC3.

```
PAGE  1        A I D S     T I C K E T     S T A T U S     A C C E S S
02/20/74
                        .HEAD   A I D S     T I C K E T     S T A T U S     A C C E S S

                        .TITLE  TSA.

          000040        SIZE=   40        ; STATUS TABLE ENTRY SIZE

                        .EXTD   TSTB      ; TICKET STATUS TABLE BEGINNING
                        .EXTD   TSTE      ; TICKET STATUS TABLE END
                        .EXTD   USTE      ; UNIT STATUS TABLE END

                        .ENT    TS1.      ; TICKET STATUS INSERT
                        .ENT    TS2.      ; TICKET STATUS DELETE
                        .ENT    TS3.      ; TICKET STATUS LOCATE

                        .NREL

                        ;       TICKET STATUS LOCATE

    00000'054413  TS3.: STA 3   RTRN      ; SAVE RETURN ADDRESS
    00001'004521        JSR     SERCH     ; LOOK FOR MATCHING ENTRY
    00002'000402        SKIP              ; NO MATCH, ERROR RETURN

    00003'010410  OUT:  ISZ     RTRN      ; FOUND IT, SUCCESS
    00004'020403        LDA 0   NAME      ; RESTORE AC0
    00005'024403        LDA 1   NUMB      ; RESTORE AC1
    00006'002405        JMP     @RTRN     ; RETURN

    00007'000000  NAME: 0                 ; CURRENT CONTROL CENTER NAME (CCN)
    00010'000000  NUMB: 0                 ; WANTED TICKET NUMBER
    00011'000000  NEW:  0                 ; ADDRESS OF NEW RECORD
    00012'000000  COUNT:0                 ; LOOP COUNTER

    00013'000000  RTRN: 0                 ; RETURN ADDRESS
    00014'000000  SRET: 0                 ; SUBROUTINE RETURN ADDRESS

                        ;       TICKET STATUS DELETE

    00015'054776  TS2.: STA 3   RTRN      ; SAVE RETURN ADDRESS
    00016'004504        JSR     SERCH     ; LOOK FOR MATCHING ENTRY
    00017'000765        JMP     OUT+1     ; NO MATCH, ERROR RETURN
    00020'020501        LDA 0   .SIZE     ; 20 WORD ENTRIES
    00021'116400        SUB     0,3       ; POINT TO NEXT ENTRY

    00022'020001$ TEST2: LDA 0  TSTB      ; GET END OF TABLE
    00023'116512        SLE     0,3       ; MORE TO MOVE?
    00024'000414        JMP     MOVED     ; NO, CONTINUE
    00025'020474        LDA 0   .SIZE     ; 20 WORD ENTRIES
    00026'040764        STA 0   COUNT     ; SET UP COUNTER

    00027'025400  NEXT2: LDA 1  0,3       ; GET TABLE WORD
    00030'045440        STA 1   SIZE,3    ; MOVE IT UP ONE RECORD
    00031'175400        INC     3,3       ; MOVE TO NEXT WORD
    00032'014764        DSZ     COUNT     ; DONE WITH THIS ENTRY?
    00033'000774        JMP     NEXT2     ; NO, DO NEXT WORD
    00034'020465        LDA 0   .SIZE
    00035'103000        ADD     0,0       ; TWO 20 WORD ENTRIES
    00036'116400        SUB     0,3       ; MOVE BACK TWO ENTRIES
    00037'000763        JMP     TEST2     ; TEST FOR END OF TABLE
```

```
                    MOVED:  LDA 0   TSTB    ; GET END OF TABLE
                            LDA 1   .SIZE   ; 20 WORD ENTRIES
                            ADD     1,0     ; ALLOW FOR DELETED ENTRY
                            STA 0   TSTB    ; UPDATE END POINTER
                            JMP     OUT     ; TAKE SUCCESS EXIT

                    ;       TICKET STATUS INSERT

                    TSI.:   STA 3   RTRN    ; SAVE RETURN ADDRESS
                            STA 0   NAME    ; SAVE CCN
                            STA 1   NUMB    ; SAVE WANTED TICKET NUMBER
                            LDA 3   TSTB    ; GET BEGINNING OF TABLE
                            LDA 0   TSTE    ; AND END OF TABLE
                            SGT     0,3     ; NULL TABLE?
                            JMP     PAST1   ; YES, DO IT NOW
                            JSR     SERCH+2 ; LOOK FOR MATCHING ENTRY
                            SKIP            ; NOT PRESENT, CONTINUE
                            JMP     OUT     ; ALREADY PRESENT, RETURN

                    PAST1:  LDA 0   .SIZE   ; 20 WORD ENTRIES
                            SUB     0,3     ; MOVE BACK ONE ENTRY
                            STA 3   NEW     ; SAVE TABLE POINTER
                            LDA 3   TSTB    ; GET END OF TABLE
                            SUB     0,3     ; MOVE END OF TABLE
                            LDA 1   USTE    ; GET END OF UNITS
                            SGE     3,1     ; WILL NEW ENTRY FIT?
                            JMP     OUT+1   ; NO, TAKE ERROR RETURN
                            STA 3   TSTB    ; UPDATE END OF TABLE POINTER

                    TEST1:  LDA 0   NEW     ; GET ADDRESS OF NEW ENTRY
                            SLT     3,0     ; HAS SPACE BEEN ALLOWED YET?
                            JMP     FILL    ; YES, INSERT NEW ENTRY
                            LDA 0   .SIZE   ; 20 WORD ENTRIES
                            STA 0   COUNT   ; SET UP COUNTER

                    NEXT1:  LDA 1   SIZE,3  ; GET WORD TO BE MOVED
                            STA 1   0,3     ; MOVE IT UP ONE RECORD
                            INC     3,3     ; POINT TO NEXT WORD
                            DSZ     COUNT   ; DONE WITH THIS ENTRY?
                            JMP     NEXT1   ; NO, MOVE NEXT WORD
                            JMP     TEST1   ; YES, TEST IF DONE WITH MOVING

                    FILL:   LDA 3   NEW     ; SET UP BASE REGISTER
                            LDA 0   .SIZE   ; 20 WORD ENTRIES
                            STA 0   COUNT   ; SET UP COUNTER
                            SUB     0,0     ; CREATE A ZERO
                            STA 0   0,3     ; CLEAR A WORD
                            INC     3,3     ; BUMP ADDRESS
                            DSZ     COUNT   ; KICK COUNT
                            JMP     .-3     ; AND REPEAT

                            LDA 3   NEW     ; RESTORE ENTRY ADDRESS
                            LDA 0   NAME    ; GET CONTROL CENTER NAME
                            STA 0   0,3     ; INSTALL IT
                            LDA 0   NUMB    ; GET TICKET NUMBER
                            STA 1   1,3     ; INSTALL IT
                            JMP     OUT     ; YES, TAKE SUCCESS RETURN
```



```
00121'000040    .SIZE:  SIZE            ; 20 WORDS PER ENTRY

                ;       TABLE SEARCHING ROUTINE

00122'040665    SERCH:  STA 0   NAME    ; SAVE CONTROL CENTER NAME
00123'044665            STA 1   NUMB    ; SAVE DESIRED UNIT NUMBER
00124'054670            STA 3   SRET    ; SAVE RETURN ADDRESS
00125'034001$           LDA 3   TSTB    ; SET UP BASE REGISTER

00126'021401    LOOP:   LDA 0   1,3     ; GET UNIT FOR THIS ENTRY
00127'106452            SLS     0,1     ; MORE TO SEARCH?
00130'000404            JMP     NOPE    ; YES, CONTINUE

00131'106415            SNE     0,1     ; IS THIS IT?
00132'010662            ISZ     SRET    ; YES, BUMP RETURN WORD
00133'002661            JMP     @SRET   ; AND RETURN

00134'020765    NOPE:   LDA 0   .SIZE   ; 20 WORD ENTRIES
00135'117001            ADD     0,3     ; POINT TO NEXT ENTRY
00136'020002$           LDA 0   TSTE    ; GET END OF TABLE
00137'116512            SLE     0,3     ; MORE TO TRY?
00140'000766            JMP     LOOP    ; YES, DO IT
00141'002653            JMP     @SRET   ; TABLE EXHAUSTED, FAIL RETURN

                .END
```

LBL	VALUE	DEF'N	REFERENCES						
	'012'	1:31	1:48	1:53	2:35	2:40	2:46	2:50	
	'123'	2:44	2:33						
	'126'	3:11	3:23						
	'4'	2:02	1:46						
	'7'	1:28	1:24	2:11	2:54	3:06			
	'11'	1:30	2:23	2:31	2:44	2:53			
	'5'	2:37	2:41						
	'27'	1:50	1:54						
	'3'	3:19	3:13						
	'3'	1:29	1:25	2:12	2:56	3:07			
	'3'	1:23	1:40	2:06	2:19	2:28	2:58		
	'7'	2:21	2:16						
	'13'	1:33	1:19	1:23	1:26	1:38	2:10		
	'122'	3:06	1:20	1:39	2:17				
	40	1:05	1:51	2:37	3:02				
	'14'	1:34	3:08	3:16	3:17	3:24			
	'70'	2:31	2:42						
	'2'	1:44	1:58						
	'5'	2:10	1:11						
	'15'	1:38	1:12						
	''	1:19	1:13						
	$X	1:07	1:44	2:02	2:05	2:13	2:24	2:29	3:09
	$X	1:43	2:14	3:21					
	$X	1:09	2:26						
	'121'	3:02	1:41	1:47	1:55	2:03	2:21	2:34	2:45
			3:19						

FLAGGED LINES: NONE

UNIT STATUS OUTPUT (USO.)

The Unit Status Output routine will output the entire contents of the unit status table on the terminal display screen. If the entire table will not fit on that portion of the screen allocated for status display, the routine will output as much as will fit and then type the flashing message "MORE?" and await further command from the Status routine. On word from the Status routine, it will continue to output the unit status table, overwriting its previous output.


```
                        .HEAD   A I D S    U N I T   S T A T U S   O U T P U T
                        .TITLE  USO.
000040          SIZE=   40      ; STATUS TABLE ENTRY SIZE
                        .EXTD   ZAP     ; DISPLAY COMMAND ENTRY
                        .EXTD   USTB    ; UNIT STATUS TABLE BEGINNING
                        .EXTD   USTE    ; UNIT STATUS TABLE END
                        .EXTD   SPOT    ; TEMPORARY CURSOR POSITION
                        .EXTD   HOME    ; CURSOR POSITION
                        .EXTD   .CVA    ; DECIMAL CONVERSION ROUTINE
                        .EXTD   TYPE    ; AIDS TEXT TYPER
                        .EXTD   SAD     ; ADDRESS OF STATUS DISPLAY
                        .EXTD   MAD     ; ADDRESS OF MESSAGE DISPLAY
                        ;       DECIMAL CONVERSION ROUTINE PARAMETER STORAGE
000047          E0=     P7              ; UNITS
000046          E1=     P6              ; TENS
000045          E2=     P5              ; HUNDREDS
000044          E3=     P4              ; THOUSANDS
000043          E4=     P3              ; TEN-THOUSANDS
                        ;       RETURN ADDRESS STORAGE
000051          RTRN=   R1              ; MAIN ROUTINE RETURN ADDRESS
                        ;       TIT STORAGE WORDS
000060          CCN=    S0              ; CONTROL CENTER NAME
000061          UNIT=   S1              ; CURRENT UNIT NUMBER
000062          LAST=   S2              ; TARGET UNIT NUMBER
000070          POINT=  T0              ; POINTER TO NEXT ELEMENT
000071          COUNT=  T1              ; LINE COUNTER
                        .ENT    USO.
                        .NREL
                        ;       INITIAL SETUP
00000'055051    USO.:   STA 3   RTRN,2  ; SAVE RETURN ADDRESS
00001'041060            STA 0   CCN,2   ; SAVE CONTROL CENTER NAME
00002'045061            STA 1   UNIT,2  ; AND CURRENT UNIT NUMBER
00003'020011$           LDA 0   MAD     ; GET ADDRESS OF MESSAGE
00004'024010$           LDA 1   SAD     ; AND BEGINNING OF STATUS DISPLAY
00005'122700            SUBS    1,0     ; COMPUTE # OF LINES
00006'041071            STA 0   COUNT,2 ; AND SAVE IT
00007'024002$           LDA 1   USTB    ; GET ADDRESS OF UNIT STATUS TABLE
00010'045070            STA 1   POINT,2 ; INITIALIZE ELEMENT POINTER
00011'020444            LDA 0   ERASE   ; SET FLAGS TO POSITION AND ERASE
00012'024011$           LDA 1   MAD     ; GET POSITION OF MESSAGE
00013'034441            LDA 3   MPOS    ; AND OFFSET
00014'167000            ADD     3,1     ; COMPUTE ADDRESS
```

```
                JSR     @ZAP,2  ; ERASE THE MESSAGE
                LDA 1   SAD     ; GET STATUS ADDRESS
                LDA 3   HMASK   ; (= -1*400)
                ADD     3,1     ; MOVE UP ONE LINE
                JSR     @ZAP,2  ; ERASE LINE ABOVE STATUS

        ;       DISPLAY ANOTHER STATUS RECORD

                LDA 1   SAD     ; ADDRESS OF LINE TO CLEAR
TOP:    LDA 0   ERASE   ; FLAG TO CLEAR A LINE
                JSR     @ZAP,2  ; POSITION CURSOR

                LDA 3   POINT,2 ; RECOVER POINTER
TEST:   LDA 0   USTE    ; GET END OF TABLE
                SLE     0,3     ; DONE?
                JMP     TEST1   ; NO, CONTINUE
                DSZ     COUNT,2 ; MORE TO CLEAR?
                JMP     NEXT    ; YES, CLEAR THEM

                SUB     0,0
                STA 0   LAST,2  ; RESET "LAST" VALUE
DONE:   LDA 1   HOME,2  ; REST POSITION
                JSR     @ZAP,2  ; RETURN CURSOR
                LDA 0   CCN,2   ; RESTORE AC0
                JMP     @RTRN,2 ; AND RETURN

TEST1:  LDA 0   UNIT,2  ; GET CURRENT UNIT
                MOV#    0,0 SNR ; IF ZERO,
                JMP     TEST3   ; TYPE ALL OF CCN

                LDA 1   1,3     ; UNIT FROM TABLE
                SNE     0,1     ; IF EQUAL,
                JMP     BUMP    ; OUTPUT THIS RECORD

TEST2:  LDA 0   .SIZE   ; TWENTY-WORD RECORDS
                ADD     0,3     ; POINT TO NEXT RECORD
                STA 3   POINT,2 ; UPDATE ADDRESS
                JMP     TEST    ; TEST END OF TABLE

.SIZE:  SIZE            ; RECORD LENGTH (20. WORDS)
MPOS:   72.             ; MESSAGE OFFSET
ERASE:  10000           ; LINE ERASE FLAGS
HMASK:  377*400         ; UPPER-BYTE MASK
LMASK:  377             ; LOWER-BYTE MASK

TEST3:  LDA 0   CCN,2   ; GET CURRENT CCN
                MOV#    0,0 SNR ; IF ZERO,
                JMP     TEST6   ; DUMP ALL STATUS

                LDA 1   0,3     ; GET CCN FROM TABLE
                LDA 3   LMASK   ; LO-BYTE MASK
                AND#    3,0 SNR ; IF NO LOW BYTE,
                JMP     TEST4
                AND#    3,1 SZR ; DUMP ALL OF CCN
                JMP     TEST5

TEST4:  LDA 3   HMASK   ; HI-BYTE MASK
                AND     3,0     ; EXTRACT TOP BYTE
```



```
                AND     3,1     ; OF BOTH NAMES

TEST5:  LDA 3   POINT,2 ; RECOVER POINTER
                SEQ     0,1     ; IF EQUAL,
                JMP     TEST2   ; OUTPUT THIS RECORD

TEST6:  LDA 0   LAST,2  ; GET LAST UNIT NUMBER
                LDA 1   1,3     ; GET THIS UNIT NUMBER
                SLT     0,1     ; THIS ONE MORE RECENT?
                JMP     TEST2   ; NO, TRY NEXT RECORD

BUMP:   DSZ     COUNT,2 ; ANY MORE SPACE?
                JMP     DUMP    ; YES, DISPLAY IT

                LDA 1   MAD     ; MESSAGE POSITION
                LDA 0   MPOS    ; AND OFFSET
                ADD     0,1     ; CREATE ADDRESS
                SUB     0,0
                JSR     @ZAP,2  ; POSITION CURSOR

                JSR     @TYPE   ; TYPE "MORE?"
                MESS
                SUB     0,0
                JMP     DONE    ; RESTORE AND RETURN

        ;       POINT TO NEXT RECORD FOR DISPLAY

NEXT:   LDA 1   SPOT,2  ; CURSOR POSITION
                LDA 0   HMASK   ; HI-MASK (= -1*400)
                AND     0,1     ; EXTRACT LINE NUMBER
                SUB     0,1     ; AND INCREMENT BY ONE
                JMP     TOP     ; GO DISPLAY THIS LINE

        ;       DISPLAY CONTROL CENTER NAME

DUMP:   STA 1   LAST,2  ; UPDATE "LAST" VALUE
                LDA 1   @POINT,2; GET THIS CENTER NAME
                ISZ     POINT,2 ; POINT TO UNIT NUMBER
                LDA 0   LMASK   ; HALF-WORD MASK
                AND     1,0 SZR ; TEST FOR ZERO BYTE
                SUB     0,0 SKP ; IF NOT, MAKE A ZERO
                LDA 0   .40     ; GET AN ASCII SPACE
                ADD     0,1     ; ADD TO CENTER NAME
                LDA 0   TWOCH
                JSR     @ZAP,2  ; DISPLAY CENTER
                LDA 1   .40     ; ASCII SPACE
                JSR     OUT     ; DISPLAY IT

        ;       DISPLAY UNIT NUMBER

                JSR     SETUP   ; EXTRACT DIGITS
                LDA 1   E3,2    ; SEND FOUR DIGITS
                JSR     SEND
                LDA 1   E2,2
                JSR     SEND
                LDA 1   E1,2
                JSR     SEND
                LDA 1   E0,2
```

```
00147'004473          JSR     SEND
00150'004505          JSR     GAP     ; SPACE AFTER FIELD

                ;     DISPLAY STATUS CODE

00151'004501          JSR     SETUP   ; EXTRACT DIGITS
00152'025044          LDA 1   E3,2    ; SEND 1ST TWO DIGITS
00153'004475          JSR     OUT
00154'025045          LDA 1   E2,2
00155'004473          JSR     OUT

00156'024505          LDA 1   .DASH   ; ASCII "-"
00157'004471          JSR     OUT     ; SEND IT

00160'025046          LDA 1   E1,2    ; SEND LAST TWO DIGITS
00161'004467          JSR     OUT
00162'025047          LDA 1   E0,2
00163'004465          JSR     OUT
00164'004471          JSR     GAP     ; SPACE AFTER FIELD

                ;     DISPLAY TIME

00165'004465          JSR     SETUP   ; EXTRACT DIGITS
00166'025044          LDA 1   E3,2    ; SEND 1ST TWO DIGITS
00167'004461          JSR     OUT
00170'025045          LDA 1   E2,2
00171'004457          JSR     OUT

00172'024472          LDA 1   COLON   ; ASCII ":"
00173'004455          JSR     OUT     ; SEND IT

00174'025046          LDA 1   E1,2    ; SEND LAST TWO DIGITS
00175'004453          JSR     OUT
00176'025047          LDA 1   E0,2
00177'004451          JSR     OUT
00200'004455          JSR     GAP     ; SPACE AFTER FIELD

                ;     DISPLAY TICKET NUMBER

00201'011070          ISZ     POINT,2 ; SKIP POST
00202'027070          LDA 1   @POINT,2; TICKET NUMBER
00203'006006$         JSR     @.CVA   ; EXTRACT DIGITS
00204'037070          LDA 3   @POINT,2; TICKET NUMBER
00205'011070          ISZ     POINT,2 ; BUMP POINTER
00206'102400          SUB     0,0     ; CREATE A ZERO
00207'025043          LDA 1   E4,2    ; SAVE DIGIT
00210'175014          MOV#    3,3 SZR ; IF TICKET ZERO,
00211'041043          STA 0   E4,2    ; DISPLAY BLANKS

00212'004430          JSR     SEND
00213'025044          LDA 1   E3,2
00214'004426          JSR     SEND
00215'025045          LDA 1   E2,2
00216'004424          JSR     SEND
00217'025046          LDA 1   E1,2
00220'004422          JSR     SEND
00221'025047          LDA 1   E0,2
00222'004420          JSR     SEND
```



```
                ;     SEND ALPHA STRING

00223'011070  ALFA:   ISZ     POINT,2 ; BUMP POINTER PAST
00224'011070          ISZ     POINT,2 ; DUTY, WATCH WORDS
00225'102120          ADCZL   0,0     ; = -2
00226'041047          STA 0   E0,2    ; RUN THRU LOOP TWICE

00227'004426  LOOP:   JSR     GAP     ; SPACE AFTER FIELD
00230'020430          LDA 0   .14     ; 12 TWO BYTE WORDS
00231'041046          STA 0   E1,2    ; SET UP COUNTER
00232'027070          LDA 1   @POINT,2; GET TWO CHARACTERS
00233'004423          JSR     TOUT    ; AND SEND THEM
00234'011070          ISZ     POINT,2 ; POINT TO NEXT WORD
00235'015046          DSZ     E1,2    ; DONE?
00236'000774          JMP     .-4     ; NO, SEND NEXT PAIR
00237'011047          ISZ     E0,2    ; RUN IT AGAIN
00240'000767          JMP     LOOP    ; YEP
00241'000655          JMP     NEXT    ; NOPE

                ;     LEADING ZERO SUPRESSION ROUTINE

00242'021043  SEND:   LDA 0   E4,2
00243'106415          SNE     0,1
00244'000403          JMP     OUT-1
00245'015043          DSZ     E4,2
00246'000402          JMP     OUT
00247'024412          LDA 1   .40
00250'020415  OUT:    LDA 0   ONECH
00251'003001$         JMP     @ZAP,2

                ;     FIELD SETUP ROUTINE

00252'027070  SETUP:  LDA 1   @POINT,2; GET NEXT FIELD VALUE
00253'011070          ISZ     POINT,2 ; POINT TO NEXT FIELD ELEMENT
00254'002006$         JMP     @.CVA   ; EXTRACT DIGITS

                ;     FIELD SEPARATION ROUTINE

00255'024405  GAP:    LDA 1   .4040   ; TWO ASCII SPACES
00256'020410  TOUT:   LDA 0   TWOCH   ; AND THEIR FLAGS
00257'003001$         JMP     @ZAP,2  ; SEND THEM

00260'000014  .14:    14      ; STRING LENGTH
00261'000040  .40:    40      ; ASCII SPACE
00262'020040  .4040:  40*401  ; TWO BYTES OF "SPACE"
00263'000055  .DASH:  "-      ; ASCII "-"
00264'000072  COLON:  ":      ; ASCII ":"
00265'000010  ONECH:  10      ; ONE CHARACTER OUTPUT FLAG
00266'000030  TWOCH:  30      ; TWO CHARACTER OUTPUT FLAG

000001        .TXTM   1

00267'020040  MESS:   .TXT    "  MORE\?"       ; MESSAGE TEXT
00270'046517
00271'051105
00272'056077
00273'000000
```

```
.END
```


SYMBOL	VALUE	DEF'N	REFERENCES						
ALEA	000223'	5:04							
BUMP	000103'	3:12	2:33						
CCN	000060	1:31	1:44	2:24	2:46				
COLON	000264'	5:48	4:29						
COUNT	000071	1:36	1:50	2:17	3:12				
DONE	000035'	2:22	3:24						
DUMP	000123'	3:36	3:13						
E0	000047	1:19	3:58	4:17	4:34	4:57	5:07	5:17	
E1	000046	1:20	3:56	4:15	4:32	4:55	5:11	5:15	
E2	000045	1:21	3:54	4:09	4:26	4:53			
E3	000044	1:22	3:52	4:07	4:24	4:51			
E4	000043	1:23	4:46	4:48	5:23	5:26			
ERASE	000055'	2:42	1:55	2:10					
GAP	000255'	5:40	4:02	4:19	4:36	5:09			
HMASK	000056'	2:43	2:03	2:57	3:29				
HOME	$X	1:11	2:22						
LAST	000062	1:33	2:21	3:07	3:36				
LMASK	000057'	2:44	2:51	3:39					
LOOP	000227'	5:09	5:18						
MAD	$X	1:15	1:47	1:56	3:15				
MESS	000267'	5:54	3:22						
MPOS	000054'	2:41	1:57	3:16					
NEXT	000116'	3:28	2:18	5:19					
OUTCH	000265'	5:49	5:29						
OUT	000253'	5:29	3:17	4:08	4:10	4:13	4:16	4:18	4:25
			4:27	4:30	4:33	4:35	5:25	5:27	
POINT	000070	1:35	1:53	2:13	2:37	3:03	3:37	3:38	4:40
			4:41	4:43	4:44	5:04	5:05	5:12	5:14
			5:34	5:35					
RTRN	000051	1:27	1:43	2:25					
SAD	$X	1:14	1:48	2:02	2:09				
SEND	000242'	5:23	3:53	3:55	3:57	4:01	4:50	4:52	4:54
			4:56	4:58					
SETUP	000252'	5:34	3:51	4:06	4:23				
SIZE	000040	1:05	2:40						
SPOT	$X	1:10	3:28						
TEST	000026'	2:14	2:38						
TEST1	000041'	2:27	2:16						
TEST2	000047'	2:35	3:05	3:10					
TEST3	000061'	2:46	2:29						
TEST4	000071'	2:57	2:53						
TEST5	000074'	3:03	2:55						
TEST6	000077'	3:07	2:48						
TOP	000023'	2:10	3:32						
TOUT	000256'	5:41	5:13						
TWOCH	000266'	5:50	3:44	5:41					
TYPE	$X	1:13	3:21						
UNIT	000061	1:32	1:45	2:27					
USR.	000000'	1:43	1:38						
USR3	$X	1:08	1:52						
USR4	$X	1:09	2:14						
ZAP	$X	1:07	2:01	2:05	2:11	2:23	3:19	3:45	5:3
			5:42						
.14	000260'	5:44	5:10						
.40	000261'	5:45	3:42	3:46	5:28				

OL	VALUE	DEF'N	REFERENCES	
[illegible]	000262'	5:46	5:40	
VA	$X	1:12	4:42	5:36
[illegible]	000263'	5:47	4:12	
[illegible]	000153'	2:40	2:35	

FLAGGED LINES: NONE

TICKET STATUS OUTPUT (TSO.)

The Ticket Status Output routine will output the entire contents of the ticket status table on the terminal display screen. If the entire table will not fit on that portion of the screen allocated for status display, the routine will output as much as will fit and then type the flashing message "MORE?" and await further command from the Status routine. On word from the Status routine, it will continue to output the ticket status table, overwriting its previous output.


```
                        .HEAD   A I D S    T I C K E T   S T A T U S   O U T P U
                        .TITLE  TSO.

          000040        SIZE=   40      ; STATUS TABLE ENTRY SIZE

                        .EXTD   ZAP     ; DISPLAY COMMAND ENTRY
                        .EXTD   TSTB    ; TICKET STATUS TABLE BEGINNING
                        .EXTD   TSTE    ; TICKET STATUS TABLE END
                        .EXTD   SPOT    ; TEMPORARY CURSOR POSITION
                        .EXTD   HOME    ; CURSOR POSITION
                        .EXTD   .CVA    ; DECIMAL CONVERSION ROUTINE
                        .EXTD   TYPE    ; AIDS TEXT TYPER
                        .EXTD   MAD     ; SCREEN MESSAGE ADDRESS
                        .EXTD   SAD     ; STATUS DISPLAY SCREEN ADDRESS

                        ;       DECIMAL CONVERSION ROUTINE PARAMETER STORAGE

          000047        E0=     P7              ; UNITS
          000046        E1=     P6              ; TENS
          000045        E2=     P5              ; HUNDREDS
          000044        E3=     P4              ; THOUSANDS
          000043        E4=     P3              ; TEN-THOUSANDS

                        ;       RETURN ADDRESS STORAGE

          000051        RTRN=   R1              ; MAIN ROUTINE RETURN ADDRESS

                        ;       TIT STORAGE WORDS

          000060        CCN=    S0              ; CONTROL CENTER NAME
          000061        TKT=    S1              ; CURRENT TICKET NUMBER
          000062        LAST=   S2              ; TARGET TICKET NUMBER

          000070        POINT=  T0              ; POINTER TO NEXT ELEMENT
          000071        COUNT=  T1              ; LINE COUNTER

                        .ENT    TSO.
                        .NREL

                        ;       INITIAL SETUP

000000'055051   TSO.:   STA 3   RTRN,2  ; SAVE RETURN ADDRESS
000001'041060           STA 0   CCN,2   ; SAVE CONTROL CENTER NAME
000002'045061           STA 1   TKT,2   ; AND CURRENT TICKET NUMBER

000003'020010$          LDA 0   MAD     ; GET MESSAGE ADDRESS
000004'024011$          LDA 1   SAD     ; AND DISPLAY ADDRESS
000005'122700           SUBS    1,0     ; COMPUTE # LINES
000006'041071           STA 0   COUNT,2 ; AND SAVE IT

000007'024003$          LDA 1   TSTE    ; GET ADDRESS OF TICKET STATUS TABLE
000010'020445           LDA 0   .SIZE   ; TWENTY WORD RECORDS
000011'106400           SUB     0,1     ; MOVE TO BEGINNING OF RECORD
000012'045070           STA 1   POINT,2 ; INITIALIZE ELEMENT POINTER

000013'020444           LDA 0   ERASE   ; SET FLAGS TO POSITION AND ERASE
000014'024010$          LDA 1   MAD     ; GET POSITION OF MESSAGE
```

```
                    LDA 3   MPOS    ; AND OFFSET
                    ADD     3,1     ; CREATE ADDRESS
                    JSR     @ZAP,2  ; ERASE THE MESSAGE
                    LDA 1   SAD     ; GET STATUS ADDRESS
                    LDA 3   HMASK   ; (= -1*400)
                    ADD     3,1     ; MOVE UP ONE LINE
                    JSR     @ZAP,2  ; ERASE LINE ABOVE STATUS

                ;   POINT TO NEXT RECORD FOR DISPLAY

                    LDA 1   SAD     ; ADDRESS OF LINE TO CLEAR
TOP:                LDA 0   ERASE   ; FLAG TO CLEAR A LINE
                    JSR     @ZAP,2  ; POSITION CURSOR

                    LDA 3   POINT,2 ; RECOVER POINTER
TEST:               LDA 0   TSTB    ; GET END OF TABLE
                    SGT     0,3     ; DONE?
                    JMP     TEST1   ; NO, CONTINUE
                    DSZ     COUNT,2 ; ANY LINES LEFT?
                    JMP     NEXT1   ; YES, CLEAR THEM

                    SUB     0,0
                    STA 0   LAST,2  ; RESET "LAST" VALUE
DONE:               LDA 1   HOME,2  ; REST POSITION
                    JSR     @ZAP,2  ; RETURN CURSOR
                    LDA 0   CCN,2   ; RESTORE AC0
                    JMP     @RTRN,2 ; AND RETURN

TEST1:              LDA 0   TKT,2   ; GET CURRENT TICKET
                    MOV#    0,0 SNR ; IF ZERO,
                    JMP     TEST3   ; TYPE ALL OF CCN

                    LDA 1   1,3     ; TICKET FROM TABLE
                    SNE     0,1     ; IF EQUAL,
                    JMP     BUMP    ; OUTPUT THIS RECORD

TEST2:              LDA 0   .SIZE   ; TWENTY WORD RECORDS
                    SUB     0,3     ; POINT TO NEXT RECORD
                    STA 3   POINT,2 ; UPDATE ADDRESS
                    JMP     TEST    ; TEST END OF TABLE

.SIZE:              SIZE    ; RECORD LENGTH (20. WORDS)
MPOS:               72.     ; MESSAGE OFFSET
ERASE:              10000   ; FLAG TO ERASE A LINE
HMASK:              377/400 ; UPPER-BYTE MASK
LMASK:              377     ; LOWER-BYTE MASK

TEST3:              LDA 0   CCN,2   ; GET CURRENT CCN
                    MOV#    0,0 SNR ; IF ZERO,
                    JMP     TEST6   ; DUMP ALL STATUS

                    LDA 1   0,3     ; CCN FROM TABLE
                    LDA 3   LMASK   ; LO-BYTE MASK
                    AND#    3,0 SNR ; IF NO LOW BYTE,
                    JMP     TEST4
                    AND#    3,1 SZR ; DUMP ALL OF CCN
                    JMP     TEST5
```



```
00073'034765  TEST4: LDA 3   HMASK   ; HI-BYTE MASK
00074'163400         AND     3,0     ; EXTRACT TOP BYTE
00075'167400         AND     3,1     ; OF BOTH NAMES

00076'035070  TEST5: LDA 3   POINT,2 ; RECOVER POINTER
00077'106414         SEQ     0,1     ; IF EQUAL,
00100'000751         JMP     TEST2   ; OUTPUT THIS RECORD

00101'021062  TEST6: LDA 0   LAST,2  ; GET LAST TICKET NUMBER
00102'025401         LDA 1   1,3     ; GET THIS TICKET NUMBER
00103'106112         SLT     0,1     ; THIS ONE MORE RECENT?
00104'000745         JMP     TEST2   ; NO, TRY NEXT RECORD

00105'015071  BUMP:  DSZ     COUNT,2 ; ANY MORE SPACE?
00106'000424         JMP     DUMP    ; YES, DISPLAY IT

00107'024010$        LDA 1   MAD     ; MESSAGE POSITION
00110'020746         LDA 0   MPOS    ; AND OFFSET
00111'107000         ADD     0,1     ; CREATE ADDRESS
00112'102400         SUB     0,0
00113'007001$        JSR     @ZAP,2  ; POSITION CURSOR

00114'006007$        JSR     @TYPE   ; TYPE "MORE?"
00115'000306'        MESS
00116'102400         SUB     0,0
00117'000720         JMP     DONE    ; RESTORE AND RETURN

                ;    POINT TO NEXT RECORD FOR DISPLAY

00120'035070  NEXT:  LDA 3   POINT,2 ; GET POINTER
00121'024734         LDA 1   .SIZE
00122'127000         ADD     1,1     ; OFFSET TWO RECORDS
00123'136400         SUB     1,3     ; MOVE TO NEXT RECORD
00124'055070         STA 3   POINT,2 ; UPDATE POINTER

00125'025004$ NEXT1: LDA 1   SPOT,2  ; CURSOR POSITION
00126'020732         LDA 0   HMASK   ; HI-MASK (= -1*400)
00127'107400         AND     0,1     ; EXTRACT LINE NUMBER
00130'106400         SUB     0,1     ; AND INCREMENT BY ONE
00131'000674         JMP     TOP     ; GO DISPLAY THIS LINE

                ;    DISPLAY CONTROL CENTER NAME  (CCN)

00132'045062  DUMP:  STA 1   LAST,2  ; UPDATE "LAST" VALUE
00133'027070         LDA 1   @POINT,2; GET THIS CENTER NAME
00134'011070         ISZ     POINT,2 ; POINT TO TICKET NUMBER
00135'020724         LDA 0   LMASK   ; HALF-WORD MASK
00136'123404         AND     1,0 SZR ; TEST FOR ZERO BYTE
00137'102401         SUB     0,0 SKP ; IF NOT, MAKE A ZERO
00140'020537         LDA 0   .40     ; GET AN ASCII SPACE
00141'107000         ADD     0,1     ; ADD TO CENTER NAME
00142'020542         LDA 0   TWOCH
00143'007001$        JSR     @ZAP,2  ; DISPLAY IT
00144'024533         LDA 1   .40     ; ASCII SPACE
00145'004521         JSR     OUT     ; DISPLAY IT

                ;    DISPLAY TICKET NUMBER
```

```
PAGE  4      A I D S    T I C K E T    S T A T U S    O U T P U T
02/20/74
                    JSR     SETUP   ; EXTRACT FIELD DIGITS
                    LDA 1   E4,2    ; SEND FOUR DIGITS
                    JSR     OUT
                    LDA 1   E3,2
                    JSR     OUT
                    LDA 1   E2,2
                    JSR     OUT
                    LDA 1   E1,2
                    JSR     OUT
                    LDA 1   E0,2
                    JSR     OUT
                    JSR     GAP     ; SPACE AFTER FIELD

                ;   DISPLAY STATUS CODE

                    JSR     SETUP   ; EXTRACT DIGITS
                    LDA 1   E3,2    ; SEND 1ST TWO DIGITS
                    JSR     OUT
                    LDA 1   E2,2
                    JSR     OUT

                    LDA 1   .DASH   ; ASCII "-"
                    JSR     OUT     ; SEND IT

                    LDA 1   E1,2    ; SEND LAST TWO DIGITS
                    JSR     OUT
                    LDA 1   E0,2
                    JSR     OUT
                    JSR     GAP     ; SPACE AFTER FIELD

                ;   DISPLAY TIME

                    JSR     SETUP   ; EXTRACT DIGITS
                    LDA 1   E3,2    ; SEND 1ST TWO DIGITS
                    JSR     OUT
                    LDA 1   E2,2
                    JSR     OUT

                    LDA 1   COLON   ; ASCII ":"
                    JSR     OUT     ; SEND IT

                    LDA 1   E1,2    ; SEND LAST TWO DIGITS
                    JSR     OUT
                    LDA 1   E0,2
                    JSR     OUT
                    JSR     GAP     ; SPACE AFTER FIELD

                ;   DISPLAY UNIT NUMBER

                    ISZ     POINT,2 ; SKIP POST
                    LDA 1   @POINT,2; UNIT NUMBER
                    ISZ     POINT,2 ; BUMP POINTER
                    MOV#    1,1 SNR ; UNIT MISSING?
                    JMP     NONE    ; YES, SHOW PRIORITY
                    JSR     @.CVA   ; EXTRACT DIGITS

                    LDA 1   E3,2    ; SEND FOUR DIGITS
                    JSR     SEND
```

```
PAGE  5      A I D S    T I C K E T    S T A T U S    O U T P U T
02/20/74
                    LDA 1   E2,2
                    JSR     SEND
                    LDA 1   E1,2
                    JSR     SEND
                    LDA 1   E0,2
                    JSR     SEND

                ;   SEND ALPHA STRING

ALFA:               ISZ     POINT,2 ; BUMP POINTER PAST
                    ISZ     POINT,2 ; LEVEL, TIME WORDS
                    ADCZL   0,0     ; = -2
                    STA 0   E0,2    ; RUN THRU LOOP TWICE

LOOP:               JSR     GAP     ; SPACE AFTER FIELD
                    LDA 0   .14     ; 12 2-BYTE WORDS
                    STA 0   E1,2    ; SET UP COUNTER
                    LDA 1   @POINT,2; GET TWO CHARACTERS
                    JSR     TOUT    ; AND SEND THEM
                    ISZ     POINT,2 ; POINT TO NEXT WORD
                    DSZ     E1,2    ; DONE?
                    JMP     .-4     ; NO, SEND NEXT PAIR
                    ISZ     E0,2    ; RUN LOOP AGAIN?
                    JMP     LOOP    ; YEP
                    JMP     NEXT    ; NOPE

                ;   DISPLAY PRIORITY (BLINKING)

NONE:               LDA 1   @POINT,2; GET PRIORITY
                    JSR     @.CVA   ; EXTRACT DIGITS
                    LDA 1   E1,2    ; FIRST DIGIT
                    MOVS    1,1     ; TO TOP BYTE
                    LDA 0   E0,2    ; SECOND DIGIT
                    ADD     0,1     ; COMBINE THE TWO
                    LDA 0   BLINK   ; FLAGS FOR TWO BLINKING BYTES
                    JSR     @ZAP,2  ; OUTPUT BLINKING PRIORITY
                    JMP     ALFA    ; SEND ALPHA STRING

                ;   LEADING ZERO SUPRESSION ROUTINE

SEND:               LDA 0   E4,2
                    SNE     0,1
                    JMP     OUT-1
                    DSZ     E4,2
                    JMP     OUT
                    LDA 1   .40
OUT:                LDA 0   ONECH
                    JMP     @ZAP,2

                ;   FIELD SETUP ROUTINE

SETUP:              LDA 1   @POINT,2; GET NEXT FIELD VALUE
                    ISZ     POINT,2 ; POINT TO NEXT FIELD ELEMENT
                    JMP     @.CVA   ; EXTRACT DIGITS

                ;   FIELD SEPARATION ROUTINE

GAP:                LDA 1   .4040   ; TWO ASCII SPACES
```

```
000274'020410  TOUT:   LDA 0   TWOCH   ; AND THEIR FLAGS
000275'003001s         JMP     @ZAP,2  ; SEND THEM

000276'000014  .14:    14      ; STRING LENGTH
000277'000040  .40:    40      ; ASCII SPACE
000300'020040  .4040:  40*401  ; TWO BYTES OF "SPACE"
000301'000055  .DASH:  "-      ; ASCII "-"
000302'000072  COLON:  ":      ; ASCII ":"
000303'000010  ONECH:  10      ; ONE CHARACTER OUTPUT FLAG
000304'000034  TWOCH:  30      ; TWO CHARACTER OUTPUT FLAG
000305'000032  BLINK:  32      ; TWO BLINKING CHARACTER OUTPUT FLAGS

       000001  .TXTM   1

000306'020040  MESS:   .TXT     "  MORE\?"       ; MESSAGE TEXT
000307'046517
000310'051105
000311'056077
000312'000000

                .END
```


SYMBOL	VALUE	DEF'N	REFERENCES						
ALFA	000230'	5:10	5:37						
BLINK	000305'	6:11	5:35						
BUMP	000105'	3:14	2:35						
CCN	000060	1:31	1:44	2:26	2:48				
COLON	000302'	6:08	4:39						
COUNT	000071	1:36	1:50	2:19	3:14				
DONE	000037'	2:24	3:26						
DUMP	000132'	3:44	3:15						
E0	000047	1:19	4:10	4:27	4:44	5:05	5:13	5:23	5:33
E1	000046	1:20	4:08	4:25	4:42	5:03	5:17	5:21	5:31
E2	000045	1:21	4:06	4:19	4:36	5:01			
E3	000044	1:22	4:04	4:17	4:34	4:57			
E4	000043	1:23	4:02	5:41	5:44				
ERASE	000057'	2:44	1:57	2:12					
GAP	000273'	5:58	4:12	4:29	4:46	5:15			
HMASK	000060'	2:45	2:05	3:01	3:37				
HOME	$X	1:11	2:24						
LAST	000062	1:33	2:23	3:09	3:44				
LMASK	000061'	2:46	2:53	3:47					
LOOP	000234'	5:15	5:24						
MAD	$X	1:14	1:47	1:58	3:17				
MESS	000306'	6:15	3:24						
MPOS	000056'	2:43	2:01	3:18					
NEXT	000120'	3:30	5:25						
NEXT1	000125'	3:36	2:20						
NOP	000247'	5:29	4:54						
ONECH	000303'	6:09	5:47						
OUT	000266'	5:47	3:55	4:03	4:05	4:07	4:09	4:11	4:18
			4:20	4:23	4:26	4:28	4:35	4:37	4:40
			4:43	4:45	5:43	5:45			
POINT	000070	1:35	1:55	2:15	2:39	3:05	3:30	3:34	3:45
			3:46	4:50	4:51	4:52	5:10	5:11	5:18
			5:20	5:29	5:52	5:53			
RTRN	000051	1:27	1:43	2:27					
SAD	$X	1:15	1:48	2:04	2:11				
SEND	000260'	5:41	4:58	5:02	5:04	5:06			
SETUP	000270'	5:52	4:01	4:16	4:33				
SIZE	000040	1:05	2:42						
SPOT	$X	1:10	3:36						
TEST	000030'	2:16	2:40						
TEST1	000043'	2:29	2:18						
TEST2	000051'	2:37	3:07	3:12					
TEST3	000062'	2:48	2:31						
TEST4	000073'	3:01	2:55						
TEST5	000076'	3:05	2:57						
TEST6	000101'	3:09	2:50						
TXT	000061	1:32	1:45	2:29					
TOP	000025'	2:12	3:40						
TOUT	000274'	6:01	5:19						
TSO.	000000'	1:43	1:38						
TST	$X	1:08	2:16						
TST	$X	1:09	1:52						
TWOCH	000304'	6:10	3:52	6:01					
TYPE	$X	1:13	3:23						
ZAP	$X	1:07	2:03	2:07	2:13	2:25	3:21	3:53	5:36

OL	VALUE	DEF'N	REFERENCES		
			5:48	6:02	
6	000276'	6:04	5:16		
A	000277'	6:05	3:50	3:54	5:46
40	000300'	6:06	5:58		
/A	$X	1:12	4:55	5:30	5:54
26	000301'	6:07	4:22		
4E	000055'	2:42	1:53	2:37	3:31

FLAGGED LINES: NONE

CONSOLE LOG WRITER (LOG.)

The Console Log Writer initiates an internal logging routine and returns to the system. The log writer is initiated by a subroutine call in the routine needing to make a log entry. The internal logging routine extracts pertinent data from the initiating terminal's information table and writes it into a log file on disk and displays it on the console device.


```
                .HEAD   A I D S     C O N S O L E     L O G     W R I T E R
                .TITLE  LOG.

                .EXTD   .CVA    ; ASCII CONVERSION
                .EXTD   .CCN    ; CENTER LOOK-UP

                .EXTD   ID      ; TERMINAL NUMBER
                .EXTD   ZAP     ; TERMINAL OUTPUT
                .EXTD   SPOT    ; CARRIAGE POSITION
                .EXTD   CCN     ; CONTROL CENTER
                .EXTD   OPR     ; OPERATOR NUMBER
                .EXTD   UNIT    ; UNIT NUMBER

000040  CODE=   P0      ; INCIDENT/STATUS
000041  POST=   P1      ; POST NUMBER
000042  EVENT=  P2      ; TICKET NUMBER

000043  D5=     P3
000044  D4=     P4
000045  D3=     P5      ; DECIMAL DIGITS
000046  D2=     P6
000047  D1=     P7

000051  RL=     R1      ; RETURN ADDRESS

000124  TEXT=   124     ; STRING OFFSET
000030  SIZE=   24.     ; STRING LENGTH

                .EXTN   CTIT    ; CONSOLE TABLE

                .ENT    LOG.,SLOB
                .NREL
```

```
                        ; INITIATE LOG PROCESS

                LOG.:   STA 3   RL,2    ; SAVE RETURN
                        SKIP            ; SKIP AROUND
                TRY:    IDLE            ; WAIT AWHILE
                        LDA 1   START   ; START ADDRESS
                        STA 1   BLK1    ; INTO IOC BLOCK
                        LDA 2   .IOC1   ; IOCB ADDRESS
                        .IO             ; START A PROCESS
                        JMP     TRY     ; RETRY ON FAILURE
                        LDA 2   2       ; RECOVER TIT
                        LDA 3   BLK1    ; NEW PCB ADDRESS
                        STA 2   PLOC,3  ; PASS TIT ADDRESS
                        JMP     @RL,2   ; THEN RETURN

                .IOC1:  .+1
                        0
                        4
                BLK1:   0
                        0
                        0

            000012  PLOC=   12          ; PROCESS LOCAL
                START:  @GO             ; PROCESS START

                        ; BEGIN LOGGING PROCESS

                GO:     LDA 2   .CTIT   ; CONSOLE TABLE
                        LDA 0   ID,2    ; TERMINAL NUMBER
                        ENQ             ; WAIT FOR IT

                        LDA 2   2       ; RECOVER TIT
                        LDA 0   ID,2    ; TERMINAL NUMBER
                        ENQ             ; WAIT FOR IT
```



```
                        ; HOURS, MINUTES, SECONDS

                        LDA 1   HOURS   ; DINOS HOURS WORD
                        JSR     @.CVA   ; MAKE ASCII
                        LDA 1   D2,2
                        JSR     OUT     ; TENS OF HOURS
                        LDA 1   D1,2
                        JSR     OUT     ; ONES OF HOURS
                        LDA 1   COLON
                        JSR     OUT     ; AND A COLON

                        LDA 1   MINS    ; DINOS MINUTES WORD
                        JSR     @.CVA   ; MAKE ASCII
                        LDA 1   D2,2
                        JSR     OUT     ; TENS OF MINUTES
                        LDA 1   D1,2
                        JSR     OUT     ; ONES OF MINUTES
                        LDA 1   COLON
                        JSR     OUT     ; AND A COLON

                        LDA 1   SECS    ; DINOS SECONDS WORD
                        JSR     @.CVA   ; MAKE ASCII
                        LDA 1   D2,2
                        JSR     OUT     ; TENS OF SECONDS
                        LDA 1   D1,2
                        JSR     OUT     ; ONES OF SECONDS

                        JSR     GAP     ; AND TWO SPACES

                    ; UNIT NUMBER

                        LDA 1   UNIT,2  ; UNIT NUMBER
                        JSR     @.CVA   ; MAKE ASCII
                        JSR     DUMP    ; DUMP 4 DIGITS

                    ; INCIDENT/STATUS CODE

                        LDA 1   CODE,2  ; CODE NUMBER
                        JSR     @.CVA   ; MAKE ASCII
                        LDA 1   D4,2
                        JSR     SEND    ; DIGIT 4
                        LDA 1   D3,2
                        JSR     SEND    ; DIGIT 3
                        LDA 1   DASH
                        JSR     SEND    ; A DASH
                        LDA 1   D2,2
                        JSR     SEND    ; DIGIT 2
                        LDA 1   D1,2
                        JSR     SEND    ; DIGIT 1

                        JSR     GAP     ; 2 SPACES
```

```
                ; TICKET NUMBER

00100'025042            LDA 1   EVENT,2 ; TICKET NUMBER
00101'006001$           JSR     @.CVA   ; MAKE ASCII
00102'025043            LDA 1   D5,2    ; FIRST DIGIT
00103'021042            LDA 0   EVENT,2 ; TICKET NUMBER
00104'101014            MOV#    0,0 SZR ; IF NOT ZERO,
00105'102401            SUB     0,0 SKP ; CLEAR FLAG,
00106'021043            LDA 0   FLAG,2  ; ELSE LEAVE IT
00107'004470            JSR     SEND+1  ; FIRST DIGIT
00110'004457            JSR     DUMP    ; THEN 4 MORE

                ; TEXT STRING

00111'020440            LDA 0   .SIZE   ; STRING LENGTH
00112'044444            STA 0   COUNT   ; BYTE COUNTER
00113'034435            LDA 3   .TEXT   ; STRING OFFSET
00114'157120            ADDZL   2,3     ; BYTE POINTER

00115'054436  LOOP:     STA 3   POINT   ; SAVE POINTER
00116'175220            MOVZR   3,3     ; MAKE ADDRESS
00117'025400            LDA 1   0,3     ; FETCH A WORD
00120'125013            MOV#    1,1 SNC ; USE LOW BYTE
00121'125300            MOVS    1,1     ; OR HIGH BYTE
00122'004467            JSR     OUT     ; OUTPUT A BYTE
00123'034431            LDA 3   POINT   ; GET POINTER
00124'175400            INC     3,3     ; AND BUMP IT
00125'021425            DSZ     COUNT   ; BUMP COUNT
00126'000767            JMP     LOOP    ; AND REPEAT

00127'004452            JSR     GAP     ; TWO SPACES

                ; POST NUMBER

00130'025041            LDA 1   POST,2  ; POST NUMBER
00131'006001$           JSR     @.CVA   ; MAKE ASCII
00132'004435            JSR     DUMP    ; DUMP 4 DIGITS

                ; DEPARTMENT NAME

00133'021006$           LDA 0   CCN,2   ; CONTROL CENTER
00134'006002$           JSR     @.CCN   ; LOCATE IN TABLE
00135'126401            SUB     1,1 SKP ; NOT IN TABLE
00136'025401            LDA 1   1,3     ; DEPARTMENT
00137'045047            STA 1   D1,2    ; SAVE LOW BYTE
00140'125300            MOVS    1,1     ; GET HIGH BYTE
00141'004443            JSR     SEND    ; AND SEND IT
00142'025047            LDA 1   D1,2    ; GET LOW BYTE
00143'004441            JSR     SEND    ; AND SEND IT
00144'004435            JSR     GAP     ; THEN 2 SPACES

                ; OPERATOR NUMBER

00145'025047$           LDA 1   OPR,2   ; OPERATOR NUMBER
00146'002001$           JMP     @.CVA   ; MAKE ASCII
                                        ; (CONTINUED)
```



```
00147'177777  .CTIT:  CTIT
00150'000124  .TEXT:  TEXT
00151'000030  .SIZE:  SIZE
00152'000000  COUNT:  0
00153'000000  POINT:  0

00154'000072  COLON:  ":
00155'000055  DASH:   "-
00156'000057  SLASH:  "/
00157'000040  SPACE:  40

00160'000015  .CR:    15
00161'000012  .LF:    12
00162'000010  .10:    10
00163'040000  OFF:    40000

00164'000331' BUFF:   SLOB
00165'000000  DRET:   0
00166'000000  SRET:   0
000043        FLAG=   D5

00167'054776  DUMP:   STA 3   DRET    ; SAVE RETURN
00170'025044          LDA 1   D4,2
00171'004413          JSR     SEND    ; DIGIT 4
00172'025045          LDA 1   D3,2
00173'004411          JSR     SEND    ; DIGIT 3
00174'025046          LDA 1   D2,2
00175'004407          JSR     SEND    ; DIGIT 2
00176'025047          LDA 1   D1,2
00177'004405          JSR     SEND    ; DIGIT 1
00200'004402          SKIP

00201'054764  GAP:    STA 3   DRET    ; SAVE RETURN
00202'004401          JSR     .+1     ; SEND ONE SPACE
00203'024754          LDA 1   SPACE   ; AND THEN ANOTHER
00204'021043  SEND:   LDA 0   FLAG,2  ; LEADING-ZERO FLAG
00205'136113          SGE     0,1     ; IF DIGIT > ZERO,
00206'102401          SUB     0,0 SKP ; RESET THE FLAG,
00207'024750          LDA 1   SPACE   ; ELSE SEND A SPACE
00210'041043          STA 0   FLAG,2  ; REPLACE THE FLAG

00211'054755  OUT:    STA 3   SRET    ; SAVE RETURN
00212'030735          LDA 2   .CTIT   ; CONSOLE TABLE
00213'020747          LDA 0   .10     ; LOW-BYTE FLAG
00214'007004$         JSR     @ZAP,2  ; TYPE ONE BYTE

00215'102520          SUBZL   0,0     ; DISK = 1
00216'004050          ENQ             ; WAIT FOR IT
00217'121000          MOV     1,0     ; GET DATA WORD
00220'030744          LDA 2   BUFF    ; DISK BUFFER
00221'034041          LDA 3   DDOS    ; SYSTEM TABLE
00222'007410          JSR     @D.OUT,3; WRITE ONE BYTE
00223'102520          SUBZL   0,0     ; DISK = 1
00224'004052          DEQ             ; GIVE IT UP

00225'030002          LDA 2   2       ; RECOVER TIT
00226'006740          JSR     @SRET   ; RETURN FROM "OUT","SEND"
00227'006736          JSR     @DRET   ; RETURN FROM "GAP","DUMP"
```

```
                        ; OPERATOR (CONTINUED)

00230'004737            JSR     DUMP    ; DUMP 4 DIGITS

                        ; MONTH, DAY, YEAR

00231'024065            LDA 1   MONTH   ; DINOS MONTH WORD
00232'006001$           JSR     @.CVA   ; MAKE ASCII
00233'025046            LDA 1   D2,2
00234'004755            JSR     OUT     ; TENS OF MONTHS
00235'025047            LDA 1   D1,2
00236'004753            JSR     OUT     ; ONES OF MONTHS
00237'024717            LDA 1   SLASH
00240'004751            JSR     OUT     ; AND A SLASH

00241'024064            LDA 1   DAY     ; DINOS DAY WORD
00242'006001$           JSR     @.CVA   ; MAKE ASCII
00243'025046            LDA 1   D2,2
00244'004745            JSR     OUT     ; TENS OF DAYS
00245'025047            LDA 1   D1,2
00246'004743            JSR     OUT     ; ONES OF DAYS
00247'024727            LDA 1   SLASH
00250'004741            JSR     OUT     ; AND A SLASH

00251'024066            LDA 1   YEAR    ; DINOS YEAR WORD
00252'006001$           JSR     @.CVA   ; MAKE ASCII
00253'025046            LDA 1   D2,2
00254'004735            JSR     OUT     ; TENS OF YEARS
00255'025047            LDA 1   D1,2
00256'004733            JSR     OUT     ; ONES OF YEARS

00257'004722            JSR     GAP     ; AND TWO SPACES

                        ; TERMINAL NUMBER

00260'025003$           LDA 1   ID,2    ; TERMINAL ID
00261'125224            MOVZR   1,1
00262'125224            MOVZR   1,1     ; DIVIDE BY 8.
00263'125224            MOVZR   1,1
00264'021043            LDA 0   FLAG,2  ; ASCII "ZERO"
00265'107401            ADD     0,1     ; MAKE DIGIT 1
00266'004723            JSR     OUT     ; AND OUTPUT

00267'025003$           LDA 1   ID,2    ; TERMINAL ID
00270'020437            LDA 0   .7      ; 3-BIT MASK
00271'107401            AND     0,1     ; MODULO 8.
00272'021043            LDA 0   FLAG,2  ; ASCII "ZERO"
00273'107401            ADD     0,1     ; MAKE DIGIT 2
00274'004715            JSR     OUT     ; AND OUTPUT

00275'024663            LDA 1   .CR
00276'004713            JSR     OUT     ; SEND RETURN
00277'024662            LDA 1   .LF
00300'004711            JSR     OUT     ; AND LINE FEED
```



```
                        ; CLEAN UP, CLEAR OUT

00301'021003$           LDA 0   ID,2    ; TERMINAL NUMBER
00302'004052            DEQ             ; GIVE IT UP

00303'102520            SUBZL   0,0     ; DISK = 1
00304'004050            ENQ             ; WAIT FOR IT
00305'030657            LDA 2   BUFF    ; DISK BUFFER
00306'034041            LDA 3   DDOS    ; SYSTEM TABLE
00307'007400            JSR     @D.CYS,3; CALL OVERLAY
00310'000012            0.EOF           ; FOR END-FILE
00311'102520            SUBZL   0,0     ; DISK = 1
00312'004052            DEQ             ; GIVE IT UP

00313'030634            LDA 2   .CTIT   ; CONSOLE TABLE
00314'020647            LDA 0   OFF     ; PRINTER-OFF
00315'025005$           LDA 1   SPOT,2  ; TO BE POLITE
00316'017004$           JSR     @ZAP,2  ; STOP PRINTER
00317'021003$           LDA 0   ID,2    ; CONSOLE ID
00320'004052            DEQ             ; GIVE IT UP

00321'024003            LDA 1   3       ; PROCESS BLOCK
00322'044406            STA 1   BLK2    ; INTO IOC BLOCK
00323'030402            LDA 2   .IOC2   ; IOCB ADDRESS
00324'006044            .IO             ; TERMINATE

00325'000326'   .IOC2:  .+1
00326'000000            0
00327'000007    .7:     7
00330'000000    BLK2:   0

00331'000407    SLOT:   .BLK    B.LEN   ; DISK BUFFER

        000060  HOURS=  60
        000061  MINS=   61              ; DINOS TIME WORDS
        000062  SECS=   62

        000064  DAY=    64
        000065  MONTH=  65              ; DINOS DATE WORDS
        000066  YEAR=   66

                .END
```

L	/ALUE	DEF'N	REFERENCES						
	'''17'	2:19	2:07	2:12					
	'0330'	7:30	7:23						
	'0164'	5:17	5:50	7:08					
	$X	1:11	4:41						
	'''040	1:15	3:38						
	0'0154'	5:07	3:09	3:18					
	'0152'	5:14	4:16	4:28					
	'X	1:30	5:01						
	'''47	1:23	3:07	3:16	3:25	3:48	4:45	4:48	5:29
			6:11	6:20	6:29				
	'''046	1:22	3:05	3:14	3:23	3:46	5:27	6:09	6:18
			6:27						
	'''045	1:21	3:42	5:25					
	'''044	1:20	3:40	5:23					
	'''043	1:19	4:05	5:20					
	'0155'	5:08	3:44						
	'0164	7:38	6:16						
	'''165'	5:18	5:22	5:33	5:58				
	0'0167'	5:22	3:34	4:11	4:37	6:03			
	0'0042	1:17	4:03	4:06					
	0'0043	5:20	4:09	5:36	5:40	6:40	6:47		
	'''241'	5:33	3:28	3:51	4:31	4:50	6:32		
	0'0023'	2:28	2:24						
	'0'063	7:34	3:03						
	$X	1:08	2:29	2:33	6:36	6:44	7:03	7:19	
	0'0029'	2:03	1:32						
	'''115'	4:23	4:29						
	'''61	7:35	3:12						
	'''065	7:39	6:07						
	'''163'	5:15	7:16						
	$X	1:12	4:54						
	'''211'	5:42	3:06	3:08	3:10	3:15	3:17	3:19	3:24
			3:26	4:25	6:10	6:12	6:14	6:19	6:21
			6:23	6:28	6:30	6:42	6:49	6:52	6:54
	'''12	2:23	2:13						
	0'0153'	5:05	4:20	4:26					
	'''041	1:16	4:35						
	'''51	1:25	2:03	2:14					
	0'0062	7:36	3:21						
	'0204'	5:36	3:41	3:43	3:45	3:47	3:49	4:10	4:47
			4:49	5:24	5:26	5:28	5:30		
	'''33	1:28	5:03						
	0'0156'	5:09	6:13	6:22					
	'''31'	7:32	1:32	5:17					
	'''57'	5:10	5:35	5:39					
	$X	1:10	7:17						
	'''6'	5:19	5:42	5:57					
	0'0022'	2:24	2:06						
	'''24	1:27	5:02						
	'''2'	2:05	2:10						
	$X	1:13	3:32						
	'0066	7:40	6:25						
	$X	1:09	5:45	7:18					
	'''2'	5:14	5:44						
	'''27'	7:29	6:45						

SYMBOL	VALUE	DEF'N	REFERENCES						
.CC	$X	1:06	4:42						
.CR	000160'	5:12	6:51						
.CTIT	000147'	5:01	2:28	5:43	7:15				
.CVA	$X	1:05	3:04	3:13	3:22	3:33	3:39	4:04	4:36
			4:55	6:08	6:17	6:26			
.IOC1	000014'	2:16	2:08						
.IOC2	000325'	7:27	7:24						
.LF	000161'	5:13	6:53						
.SIZE	000151'	5:03	4:15						
.TEXT	000150'	5:02	4:17						

FLAGGED LINES: NONE

TEXT STRING OUTUT (TYPE.)

The Text String Output routine is used to output strings of ASCII text on the terminal output display. It is passed the address of the string address in AC3 and outputs through the output driver routine entry in the terminal information table (ZAP). The string is packed two characters in a word, the 1st character in the upper byte, and is terminated with a byte of binary zeroes. The routine increments AC3 for use as its return address.

Example:

```
        .
        .
        .
JSR @.TYPE  ; CALL TO TYPE.
STADR       ; ADDRESS OF STRING
            ; ROUTINE RETURNS HERE

        .
        .
        .
```



```
1                       .HEAD   A I D S     T E X T   S T R I N G   O U T P U T
2
3                       .TITLE  TYPE.
4
5                       .EXTD   ZAP         ; DISPLAY COMMAND ENTRY
6
7           000070      RT=     T0          ; RETURN ADDRESS
8           000071      POINT=  T1          ; DATA BYTE POINTER
9
10                      .ENT    TYPE.
11                      .NREL
12
13    000000'021400     TYPE.:  LDA 0   0,3     ; GET ADDRESS
14    000001'101120             MOVZL   0,0     ; MAKE BYTER
15    000002'041071             STA 0   POINT,2 ; AND SAVE IT
16    000003'175400             INC     3,3     ; BUMP RETURN
17    000004'055070             STA 3   RT,2    ; AND SAVE IT
18
19    000005'035071     LOOP:   LDA 3   POINT,2 ; GET BYTER
20    000006'175220             MOVZR   3,3     ; MAKE ADDRESS
21    000007'025400             LDA 1   0,3     ; GET A DATA WORD
22    000010'125302             MOVS    1,1 SZC ; TEST ADDRESS
23    000011'125300             MOVS    1,1     ; IF EVEN, SWAP
24
25    000012'020410             LDA 0   .377
26    000013'107405             AND     0,1 SNR ; EXTRACT DATA
27    000014'003070             JMP     @RT,2   ; IF ZERO, RETURN
28
29    000015'020404             LDA 0   .10
30    000016'037001$            JSR     @ZAP,2  ; TYPE A CHARACTER
31    000017'011071             ISZ     POINT,2 ; BUMP BYTER
32    000020'000765             JMP     LOOP    ; AND REPEAT
33
34    000021'000010     .10:    10
35    000022'000377     .377:   377
36
37                      .END
```

A I D S TEXT STRING OUTPUT

AL	VALUE	DEF'N	REFERENCES		
[illegible]	[illegible]5'	1:19	1:32		
[illegible]	[illegible]71	1:08	1:15	1:19	1:31
[illegible]	[illegible]73	1:07	1:17	1:27	
[illegible]	[illegible]0'	1:13	1:10		
[illegible]	$X	1:05	1:30		
[illegible]	[illegible]21'	1:34	1:29		
[illegible]	[illegible]22'	1:35	1:25		

FLAGGED LINES: NONE

DATA STRING INPUT (ASK.)

The data string input routine reads an input data string from the terminal keyboard and stores it starting at the address in AC1. The maximum number of characters allowed in the input string is passed in AC0. If the input string is empty, a direct return is taken. If the input string is not empty, a skip return is taken.


```
                        .HEAD   A I D S    D A T A    S T R I N G    I N P U T
                        .TITLE  ASK.
                        .EXTD   ZAP     ; OUTPUT ENTRY VECTOR
                        .EXTD   EDIT    ; EDIT TABLE ADDRESS
                        .EXTD   CHAR    ; INPUT DATA CHARACTER
                        .EXTD   SPOT    ; CURSOR POSITION
          000053    RA=     R3
          000050    RC=     R0
          000067    POINT=  S7
          000066    COUNT=  S6
          000065    LIMIT=  S5
          000064    EMPTY=  S4
          000063    FILL=   S3
                        .ENT    ASK.
                        .NREL
00000'055053    ASK.:   STA     3,RA,2          ; SAVE RETURN
00001'135120            MOVZL   1,3             ; MAKE POINTER
00002'101112            MOVL#   0,0 SZC         ; IF COUNT < 0,
00003'100401            NEG     0,0 SKP         ; MAKE POSITIVE
00004'126401            SUB     1,1 SKP         ; GET ZEROES
00005'024507            LDA     1,BLANK         ; OR BLANKS
00006'045063            STA     1,FILL,2        ; AND STORE

00007'041065            STA     0,LIMIT,2       ; SAVE LIMIT
00010'055067            STA     3,POINT,2       ; AND POINTER
00011'100641            NEGOR   0,0             ; NEGATE COUNT
00012'175220            MOVZR   3,3             ; MAKE ADDRESS
00013'045400            STA     1,0,3           ; INSERT FILLER
00014'175400            INC     3,3             ; BUMP ADDRESS
00015'101404            INC     0,0,SZR         ; AND COUNT
00016'000775            JMP     .-3             ; AND REPEAT

00017'041066            STA     0,COUNT,2       ; CLEAR COUNT
00020'041064            STA     0,EMPTY,2       ; AND SET EMPTY
00021'020501            LDA     0,.130
00022'024471            LDA     1,COLON         ; TYPE ": "
00023'007001$           JSR     @ZAP,2
00024'014413            JSR     SEDIT           ; PRIME AC3

00025'000075'           BACK    ; BACKSPACE
00026'000126'           TAB     ; TAB
00027'000104'           REV     ; REVERSE
00030'000123'           BELL
00031'000123'           BELL
00032'000143'           END     ; RETURN
00033'000123'           BELL
00034'000123'           BELL
00035'000041'           SPACE   ; SPACE
00036'000044'           DATA    ; DATA

00037'055002$   SEDIT:  STA     3,EDIT,2        ; SET EDIT WORD
00040'000351'           JMP     @RC,2           ; RESUME AWAIT
```

```
                        ; SPACE CHARACTER

00041 021064  SPACE:  LDA     0,EMPTY,2
00042 101015          MOV#    0,0,SNR         ; IF FIELD EMPTY,
00043 001400          JMP     0,3             ; IGNORE SPACES

                        ; DATA CHARACTER

00044 025066  DATA:   LDA     1,COUNT,2       ; GET DATA COUNT
00045 021065          LDA     0,LIMIT,2       ; AND COUNT LIMIT
00046 106513          SGT     0,1             ; IF PAST LIMIT,
00047 000454          JMP     BELL            ; RING THE BELL
00050 055051          STA     3,RC,2          ; SAVE RETURN
00051 135400          INC     1,3             ; BUMP DATA COUNT
00052 055066          STA     3,COUNT,2       ; AND SAVE IT
00053 021064          LDA     0,EMPTY,2       ; LAST VALID DATA
00054 116113          SGE     0,3
00055 055064          STA     3,EMPTY,2       ; UPDATE LAST DATA

00056 035067          LDA     3,POINT,2       ; GET STRING BYTER
00057 137000          ADD     1,3             ; ADD ON DATA COUNT
00060 021003$         LDA     0,CHAR,2        ; GET INPUT DATA
00061 024434          LDA     1,.377          ; HALF-WORD MASK
00062 175223          MOVZR   3,3,SNC         ; MAKE ADDRESS
00063 123400          MOVS    0,0,SKP         ; SHIFT DATA OR MASK
00064 124000          COM     1,1

00065 031400          LDA     2,0,3           ; GET DATA WORD
00066 133400          AND     1,2             ; EXTRACT HALF
00067 113000          ADD     0,2             ; ADD THE REST
00070 051400          STA     2,0,3           ; PUT IT BACK
00071 043002          LDA     2,2             ; RESTORE AC2
00072 055051          LDA     3,RC,2          ; RECOVER RETURN
00073 025003$         LDA     1,CHAR,2        ; GET DATA
00074 000433          JMP     OUT             ; ECHO IT

                        ; BACKSPACE

00075 025066  BACK:   LDA     1,COUNT,2       ; GET DATA COUNT
00076 124405          NEG     1,1,SNR         ; IF ZERO,
00077 000424          JMP     BELL            ; RING THE BELL
00100 124000          COM     1,1             ; REDUCE COUNT
00101 045066          STA     1,COUNT,2       ; AND SAVE IT
00102 024415          LDA     1,.10           ; GET BACKSPACE
00103 000421          JMP     OUT             ; AND SEND IT

                        ; REVERSE TAB

00104 025004$ REV:    LDA     1,SPOT,2        ; CURSOR POSITION
00105 021066          LDA     0,COUNT,2       ; STRING LENGTH
00106 106400          SUB     0,1             ; INITIAL POSITION
00107 102400          SUB     0,0
00110 041066          STA     0,COUNT,2       ; CLEAR DATA COUNT
00111 024427          LDA     0,.100
00112 003001$         JMP     @ZAP,2          ; REPOSITION

00113 035044  COLON:  ":*400+"
```



```
00114 020040  BLANK:  " *401
00115 000377  .377:   377
00116 000007  .7:     7
00117 000010  .10:    10
00120 000100  .100:   100
00121 000110  .110:   110
00122 010130  .130:   10130

                        ; RING BELL FOR ERROR

00123 024773  BELL:   LDA     1,.7            ; ASCII BELL-CODE
00124 020775  OUT:    LDA     0,.110
00125 003001$         JMP     @ZAP,2          ; EXIT THRU OUTPUT

                        ; TERMINATE

00126 055050  TAB:    STA     3,RC,2          ; SAVE RETURN
00127 025004$         LDA     1,SPOT,2        ; CURSOR POSITION
00130 021066          LDA     0,COUNT,2       ; STRING LENGTH
00131 106400          SUB     0,1             ; INITIAL POSITION
00132 021064          LDA     0,EMPTY,2       ; GET MAX. LENGTH
00133 107000          ADD     0,1             ; MOVE PAST DATA

00134 176400          SUB     3,3
00135 055002$         STA     3,EDIT,2        ; CLEAR EDIT TABLE
00136 035053          LDA     3,RA,2          ; RETURN ADDRESS
00137 101014          MOV#    0,0,SZR         ; IF ANY DATA,
00140 175400          INC     3,3             ; SKIP RETURN
00141 020757          LDA     0,.100          ; POSITION FLAGS
00142 003001$         JMP     @ZAP,2          ; EXIT THRU ZAP

00143 055050  END:    STA     3,RC,2          ; SAVE RETURN
00144 025066          LDA     1,COUNT,2       ; GET DATA COUNT
00145 045064          STA     1,EMPTY,2       ; SET EMPTY FLAG
00146 021065  ERASE:  LDA     0,LIMIT,2       ; AND COUNT LIMIT
00147 122112          SLT     1,0             ; IF PAST LIMIT,
00150 000757          JMP     TAB+1           ; ALL DONE

00151 035067          LDA     3,POINT,2       ; GET STRING BYTER
00152 137000          ADD     1,3             ; DATA POSITION
00153 024742          LDA     1,.377          ; HALF-WORD MASK
00154 021063          LDA     0,FILL,2        ; FILL CHARACTER
00155 175222          MOVZR   3,3,SZC         ; MAKE ADDRESS
00156 124000          COM     1,1             ; AND SHIFT MASK
00157 123000          ANDS    1,0             ; EXTRACT A FILL

00160 031400          LDA     2,0,3           ; GET A DATA WORD
00161 133400          AND     1,2             ; EXTRACT DATA
00162 113000          ADD     0,2             ; AND ADD FILL
00163 051400          STA     2,0,3           ; REPLACE DATA
00164 043002          LDA     2,2             ; RESTORE AC2
00165 011066          ISZ     COUNT,2         ; BUMP DATA COUNT
00166 024726          LDA     1,BLANK         ; ASCII SPACE-CODE
00167 004735          JSR     OUT             ; ERASE ONE SPACE
00170 025066          LDA     1,COUNT,2       ; GET DATA COUNT
00171 000755          JMP     ERASE           ; REPEAT

                      .END
```

SYMBOL	VALUE	DEF'N	REFERENCES						
..	000000'	1:22	1:19						
[illegible]	[illegible]75'	2:40	1:46						
LL	000123'	3:11	1:49	1:50	1:52	1:53	2:13	2:42	
[illegible]K	000114'	3:01	1:27	3:53					
R	$X	1:07	2:23	2:35					
[illegible]OT	000113'	2:58	1:42						
JET	000065	1:14	1:39	2:10	2:16	2:40	2:44	2:51	2:54
			3:19	3:33	3:52	3:55			
[illegible]A	000044'	2:10	1:55						
IT	$X	1:06	1:57	3:25					
[illegible]RY	000064	1:16	1:40	2:04	2:17	2:19	3:21	3:34	
	[illegible]143'	3:32	1:51						
[illegible]F	000146'	3:35	3:56						
LL	000063	1:17	1:28	3:42					
[illegible]IT	000065	1:15	1:30	2:11	3:35				
[illegible]	000124'	3:12	2:36	2:46	3:54				
[illegible]T	[illegible]67	1:13	1:31	2:21	3:39				
	[illegible]053	1:10	1:22	3:26					
	[illegible]61	1:11	1:58	2:14	2:34	3:17	3:32		
[illegible]	000104'	2:50	1:48						
IT	000037'	1:57	1:44						
[illegible]E	000041'	2:04	1:54						
	$X	1:08	2:50	3:18					
	[illegible]126'	3:17	1:47	3:37					
	$X	1:05	1:43	2:56	3:13	3:30			
[illegible]	[illegible]117'	3:04	2:45						
[illegible]	[illegible]120'	3:05	2:55	3:29					
[illegible]N	000121'	3:06	3:12						
[illegible]N	000122'	3:07	1:41						
[illegible]T	[illegible]115'	3:02	2:24	3:41					
[illegible]	[illegible]116'	3:03	3:11						

FLAGGED LINES: NONE

FORMAT READ (GET.)

The Format Read routine is used to read a character from a data block. The data character is returned in AC1. The format block address, data block address, format field number and character position in the field are passed in the terminal information table.


```
                        .HEAD   A I D S     F O R M A T     R E A D
                        .TITLE  GET.
                        .EXTD   FORM    ; FORMAT BLOCK ADDRESS
                        .EXTD   DATA    ; DATA BLOCK ADDRESS
                        .EXTD   FIELD   ; FORMAT FIELD NUMBER
                        .EXTD   INDEX   ; CHARACTER POSITION IN FIELD
000070          CHAR=   T0              ; DATA CHARACTER STORAGE
000071          RTRN=   T1              ; RETURN ADDRESS STORAGE
000072          RADR=   T2              ; RECORD ADDRESS STORAGE
                        .ENT    GET.
                        .NREL
000000'055071   GET.:   STA 3   RTRN,2  ; SAVE RETURN ADDRESS
000001'021003$          LDA 0   FIELD,2 ; PICK UP FIELD NUMBER
000002'103120           ADDZL   0,0     ; *4 ( FOUR WORD RECORDS )
000003'035001$          LDA 3   FORM,2  ; GET FORMAT BLOCK ADDRESS
000004'035407           LDA 3   T.SA,3  ; FORMAT START ADDRESS
000005'117003           ADD     0,3     ; COMPUTE RECORD ADDRESS
000006'055072           STA 3   RADR,2  ; SAVE IT
                        .
000007'021403           LDA 0   3,3     ; FETCH DATA WORD
000010'024461           LDA 1   .177    ; PICK UP MASK
000011'107400           AND     0,1     ; EXTRACT FIELD LENGTH
000012'021004$          LDA 0   INDEX,2 ; GET POSITION IN FIELD
000013'106042           ADCO    0,1 SZC ; PAST END OF FIELD?
000014'000447           JMP     SPACE   ; YES, EXIT

000015'021403           LDA 0   3,3     ; FETCH DATA WORD
000016'035002$          LDA 3   DATA,2  ; GET DATA BLOCK ADDRESS
000017'101300           MOVS    0,0     ; MOVE WORD ADDRESS TO LOWER BYTE
000020'105100           MOVL    0,1     ; SET CARRY TO "1ST BYTE" BIT
000021'101100           MOVL    0,0     ; BYTE ADDRESS IN LOWER 9 BITS
000022'024451           LDA 1   .777    ; FETCH MASK
000023'107400           AND     0,1     ; EXTRACT BYTE ADDRESS

000024'021004$          LDA 0   INDEX,2 ; GET POSITION IN FIELD
000025'175120           MOVZL   3,3     ; OFFSET DATA BLOCK ADDRESS
000026'117000           ADD     0,3     ; MOVE PAST PREVIOUS CHARACTERS
000027'137220           ADDZR   1,3     ; COMPUTE WORD ADDRESS OF DATA
000030'025400           LDA 1   0,3     ; PICK UP DATA WORD
000031'125003           MOV     1,1 SNC ; WHICH BYTE?
000032'125300           MOVS    1,1     ; TOP, SWAP

000033'020437           LDA 0   .377    ; FETCH MASK
000034'107400           AND     0,1     ; EXTRACT DATA CHARACTER
000035'045070           STA 1   CHAR,2  ; SAVE IT

000036'011004$          ISZ     INDEX,2 ; BUMP POSITION IN FIELD
000037'035072           LDA 3   RADR,2  ; FETCH RECORD ADDRESS
000040'025403           LDA 1   3,3     ; GET DATA WORD
000041'024437           LDA 0   .177    ; PICK UP MASK
000042'123400           AND     1,0     ; GET FIELD LENGTH
000043'025004$          LDA 1   INDEX,2 ; GET POSITION IN FIELD
000044'122443           ADCO    1,0 SNC ; PAST FIELD?
```

```
                        JMP     OUT     ; NO, EXIT

                        LDA I   0,3     ; PICK UP FLAG WORD
                        LDA 0   DONEB   ; FETCH TEST MASK
                        AND     1,0 SZR ; LAST FIELD?
                        JMP     OUT     ; YES, EXIT

                        LDA I   4,3     ; PICK UP FLAGS FOR NEXT FIELD
                        LDA 0   CONBT   ; FETCH TEST MASK
                        AND     1,0 SNR ; CONTINUATION FIELD?
                        JMP     OUT     ; NO, EXIT

                        ISZ     FIELD,2 ; MOVE TO NEXT FIELD
                        SUB     0,0     ; CREATE A ZERO
                        STA 0   INDEX,2 ; POINT TO 1ST POSITION IN FIELD

                OUT:    LDA I   CHAR,2  ; PICK UP DATA CHARACTER
                        JMP     @RTRN,2 ; RETURN

                SPACE:  ISZ     INDEX,2 ; BUMP INDEX
                        LDA I   .40     ; GET A SPACE
                        JMP     @RTRN,2 ; AND RETURN

                CONBT:  4000            ; MASK FOR CONTINUE BIT
                DONEB:  100000          ; MASK FOR LAST FIELD BIT

                .40:    40              ; ASCII SPACE
                .177:   177             ; 7-BIT MASK
                .377:   377             ; LOW BYTE MASK
                .777:   777             ; 9-BIT MASK

                .END
```


SYMBOL	VALUE	DEF'N	REFERENCES					
CHAR	000070	1:10	1:50	2:17				
CONBT	000066'	2:24	2:09					
DATA	$X	1:06	1:33					
DONEB	000067'	2:25	2:04					
FIELD	$X	1:07	1:18	2:13				
FORM	$X	1:05	1:20					
GET.	000000'	1:17	1:14					
INDEX	$X	1:08	1:28	1:40	1:52	1:57	2:15	2:20
OUT	000061'	2:17	2:01	2:06	2:11			
RADR	000012	1:12	1:23	1:53				
RTRN	000071	1:11	1:17	2:18	2:22			
SPACE	000063'	2:20	1:30					
.177	000071'	2:28	1:26	1:55				
.377	000072'	2:29	1:48					
.40	000070'	2:27	2:21					
.777	000073'	2:30	1:37					

FLAGGED LINES: NONE

FORMAT WRITE (PUT.)

The Format Write routine is used to write a character into a data block. The input data character is passed in AC1. The format block address, data block address, format field number and character position in the field are passed in the terminal information table.


```
                        .HEAD   A I D S     F O R M A T     W R I T E
                        .TITLE  PUT.
                        .EXTD   FORM    ; FORMAT BLOCK ADDRESS
                        .EXTD   DATA    ; DATA BLOCK ADDRESS
                        .EXTD   FIELD   ; FORMAT FIELD NUMBER
                        .EXTD   INDEX   ; CHARACTER POSITION IN FIELD
                        .EXTD   MOD     ; DATA BLOCK UPDATE FLAG

        000070  CHAR=   T0              ; DATA CHARACTER STORAGE
        000071  RTRN=   T1              ; RETURN ADDRESS STORAGE
        000072  RADR=   T2              ; RECORD ADDRESS STORAGE

                        .ENT    PUT.
                        .NREL
00000'020475  PUT.:   LDA 0   .177    ; FETCH MASK
00001'107400          AND     0,1     ; EXTRACT DATA BYTE
00002'045070          STA 1   CHAR,2  ; SAVE DATA CHARACTER
00003'055071          STA 3   RTRN,2  ; SAVE RETURN ADDRESS
00004'021003$         LDA 0   FIELD,2 ; PICK UP FIELD NUMBER
00005'103120          ADDZL   0,0     ; *4 ( FOUR WORD RECORDS )
00006'035001$         LDA 3   FORM,2  ; GET FORMAT BLOCK ADDRESS
00007'035407          LDA 3   T.SA,3  ; FORMAT START ADDRESS
00010'117000          ADD     0,3     ; COMPUTE RECORD ADDRESS
00011'055072          STA 3   RADR,2  ; SAVE IT

00012'021403          LDA 0   3,3     ; FETCH DATA WORD
00013'024462          LDA 1   .177    ; PICK UP MASK
00014'107400          AND     0,1     ; EXTRACT FIELD LENGTH
00015'021004$         LDA 0   INDEX,2 ; GET POSITION IN FIELD
00016'106042          ADCO    0,1 SZC ; PAST END OF FIELD?
00017'000452          JMP     OUT     ; YES, EXIT

00020'021403          LDA 0   3,3     ; FETCH DATA WORD
00021'035002$         LDA 3   DATA,2  ; GET DATA BLOCK ADDRESS
00022'126000          ADC     1,1
00023'045405$         STA 1   MOD,3   ; SET UPDATE FLAG
00024'101300          MOVS    0,0     ; MOVE WORD ADDRESS TO LOWER BYTE
00025'105100          MOVL    0,1     ; SET CARRY TO "1ST BYTE" BIT
00026'101100          MOVL    0,0     ; BYTE ADDRESS IN LOWER 9 BITS
00027'024447          LDA 1   .777    ; FETCH MASK
00030'107400          AND     0,1     ; EXTRACT BYTE ADDRESS

00031'021004$         LDA 0   INDEX,2 ; GET POSITION IN FIELD
00032'175120          MOVZL   3,3     ; OFFSET DATA BLOCK ADDRESS
00033'117000          ADD     0,3     ; MOVE PAST PREVIOUS CHARACTERS
00034'137220          ADDZR   1,3     ; COMPUTE WORD ADDRESS OF DATA
00035'025400          LDA 1   0,3     ; PICK UP DATA WORD
00036'125003          MOV     1,1 SNC ; WHICH BYTE?
00037'125300          MOVS    1,1     ; TOP, SWAP

00040'020437          LDA 0   HMASK   ; FETCH MASK
00041'107400          AND     0,1     ; REMOVE LOWER BYTE
00042'021070          LDA 0   CHAR,2  ; PICK UP DATA CHARACTER
00043'123003          ADD     1,0 SNC ; SWAP AGAIN?
00044'101300          MOVS    0,0     ; SWAP AGAIN!
```

```
                        STA Ø   Ø,3     ; INSTALL NEW DATA WORD

                        ISZ     INDEX,2 ; BUMP POSITION IN FIELD
                        LDA 3   RADR,2  ; FETCH RECORD ADDRESS
                        LDA 1   3,3     ; GET DATA WORD
                        LDA Ø   .177    ; PICK UP MASK
                        AND     1,Ø     ; GET FIELD LENGTH
                        LDA 1   INDEX,2 ; GET POSITION IN FIELD
                        ADCO    1,Ø SNC ; PAST FIELD?
                        JMP     OUT     ; NO, EXIT

                        LDA 1   Ø,3     ; PICK UP FLAG WORD
                        LDA Ø   DONEB   ; FETCH TEST MASK
                        AND     1,Ø SZR ; LAST FIELD?
                        JMP     OUT     ; YES, EXIT

                        LDA 1   4,3     ; PICK UP FLAGS FOR NEXT FIELD
                        LDA Ø   CONBT   ; FETCH TEST MASK
                        AND     1,Ø SNR ; CONTINUATION FIELD?
                        JMP     OUT     ; NO, EXIT

                        ISZ     FIELD,2 ; MOVE TO NEXT FIELD
                        SUB     Ø,Ø     ; CREATE A ZERO
                        STA Ø   INDEX,2 ; POINT TO 1ST POSITION IN FIELD

                OUT:    LDA 1   CHAR,2  ; PICK UP DATA CHARACTER
                        JMP     @RTRN,2 ; RETURN

                CONBT:  4000            ; MASK FOR CONTINUE BIT
                DONEB:  100000          ; MASK FOR LAST FIELD BIT

                .177:   177             ; 7-BIT MASK
                ./77:   777             ; 9-BIT MASK
                HMASK:  177400          ; HIGH BYTE MASK

                .END
```


SYMBOL	VALUE	DEF'N	REFERENCES				
CHAR	000070	1:11	1:20	1:56	2:26		
CONBT	000073'	2:29	2:18				
DATA	$X	1:06	1:37				
DONEB	000074'	2:30	2:13				
FIELD	$X	1:07	1:22	2:22			
FORM	$X	1:05	1:24				
HMASK	000077'	2:34	1:54				
INDEX	$X	1:08	1:32	1:46	2:03	2:08	2:24
MOD	$X	1:09	1:39				
OUT	000071'	2:26	1:34	2:10	2:15	2:20	
PUT.	000000'	1:18	1:15				
RADR	000072	1:13	1:27	2:04			
RTRN	000071	1:12	1:21	2:27			
.177	000075'	2:32	1:18	1:30	2:06		
./77	000076'	2:33	1:43				

FLAGGED LINES: NONE

FORMAT DISPLAY (DUMP.)

The Format Display routine outputs a data block to a terminal display under control of a format block. The format block address and data block address are passed in the terminal information table, as is the address of the output routine. The data is normally displayed starting on the 1st line of the output device, however by skipping the 1st word of the routine, a starting line number may be passed in ACØ.

```
PAGE  1          A I D S     F O R M A T     D I S P L A Y
02/20/74
  1                       .HEAD   A I D S     F O R M A T     D I S P L A Y
  2
  3                       .TITLE  DUMP.
  4
  5                       ;       TIT STORAGE WORD ASSIGNMENTS
  6
  7                       .EXTD   ZAP     ; DATA OUTPUT ENTRY
  8                       .EXTD   HOME    ; CURSOR REST POSITION
  9                       .EXTD   DATA    ; DATA BLOCK ADDRESS
 10                       .EXTD   FORM    ; FORMAT BLOCK ADDRESS
 11
 12              000061   START=  S1
 13              000062   FWADR=  S2
 14              000063   MAXLN=  S3
 15              000064   IFLAG=  S4
 16              000065   OFLAG=  S5
 17              000066   POINT=  S6
 18              000067   COUNT=  S7
 19
 20                       ;       BIT MASK ASSIGNMENTS
 21
 22              100000   BIT0=   100000
 23              010000   BIT3=   10000
 24              002000   BIT5=   2000
 25              001000   BIT6=   1000
 26              000400   BIT7=   400
 27              000020   BIT11=  20
 28              000010   BIT12=  10
 29              000001   BIT15=  1
 30
 31              000070   RTRN=   T0      ; MAIN RETURN ADDRESS
 32              000071   SRTN=   T1      ; SUBROUTINE RETURN ADDRESS
 33
 34                       .ENT    DUMP.
 35                       .NREL
 36
 37  00000'102400 DUMP.:  SUB     0,0     ; CREATE A ZERO
 38  00001'041061         STA     0,START,2       ; SET STARTING LINE
 39  00002'055070         STA     3,RTRN,2        ; STORE RETURN ADDRESS
 40
 41                       ;       COMPUTE POSITION
 42
 43  00003'035004$        LDA     3,FORM,2        ; PICK UP FORMAT BLOCK ADDRESS
 44  00004'035407         LDA     3,T.SA,3        ; START OF FORMAT INFORMATION
 45  00005'055062         STA     3,FWADR,2       ; INITIALIZE FORMAT WORD POINTER
 46  00006'024411         LDA     1,.1774 ; = -400
 47  00007'045063         STA     1,MAXLN,2       ; SET MAX. LINE NO.
 48
 49  00010'023062 DOIT:   LDA     0,@FWADR,2      ; GET INPUT FLAG WORD
 50  00011'041064         STA     0,IFLAG,2       ; SAVE IT
 51  00012'011062         ISZ     FWADR,2 ; BUMP FORMAT POINTER
 52  00013'023062 KO:     LDA     0,@FWADR,2      ; GET POSITION WORD
 53  00014'041066         STA     0,POINT,2       ; AND SAVE IT
 54  00015'034402         LDA     3,.1774 ; HI-BYTE MASK (=-400)
 55  00016'163401         AND     3,0,SKP ; EXTRACT LINE NUMBER
 56
 57  00017'177400 .1774:  377*400 ; HI-BYTE MASK (=-400)
 58
```

```
                LDA     1,MAXLN,2       ; GET CURRENT MAXIMUM
                SGT     0,1     ; IF LESS OR EQUAL,
                JMP     OK      ; WE'VE BEEN HERE BEFORE
                SUB     3,1 SNR ; ADVANCE LINE NUMBER
                MOV     0,1     ; IF ZERO, REPLACE IT
                STA     1,MAXLN,2       ; AND UPDATE IT
                LDA 0   START,2 ; GET LINE ORIGIN
                ADD     0,1     ; ACTUAL LINE NUMBER
                LDA     0,ELBIT ; GET LINE-ERASE FLAGS
                JSR     @ZAP,2  ; CLEAR THIS LINE
                JMP     KO      ; AND REPEAT

OK:             ISZ     FWADR,2 ; BUMP FORMAT POINTER
                LDA     0,@FWADR,2      ; GET HEADER WORD
                LDA     3,.177  ; GET MASK
                AND     0,3     ; GET HEADER BYTE COUNT IN AC3
                LDA     0,IFLAG,2       ; PICK UP FLAG WORD
                LDA     1,.3    ; GET MASK
                AND#    0,1,SZR ; IS THERE A TERM CHARACTER?
                INC     3,3     ; YES, ALLOW SPACE FOR IT
                MOVR    0,0     ; SHIFT FLAG WORD
                MOVR    0,0,SKP ; TWICE

.3:             3               ; 2 BIT MASK

                AND     1,0     ; EXTRACT SPACE COUNT
                ADD     0,3     ; TOTAL LENGTH OF HEADER IN AC3

                ;       SEND POSITION

                LDA     0,POINT,2       ; GET POSITION OF FIRST DATA CHAR
                SUB     3,0     ; MOVE TO FIRST POSITION OF HEADER
                LDA     1,START,2       ; GET STARTING POSITION
                ADD     0,1     ; PUT POSITION IN AC1
                SUB     0,0     ; SET FLAGS FOR POSITION
                JSR     @ZAP,2  ; INITIALIZE POSITION

                ;       SET UP FLAGS FOR HEADER

                LDA     1,IFLAG,2       ; GET FLAG WORD
                LDA     0,HTBIT ; PICK UP MASK
                AND     1,0,SZR ; DISPLAY HEADER BRIGHT?
                SUB     0,0,SKP ; NO, CLEAR TAG BIT
                LDA     0,DIMIT ; YES, SET TAG BIT
                LDA     1,LBYTE ; GET LOWER BYTE FLAG BIT
                ADD     1,0     ; SET LOWER BYTE FLAG
                STA     0,OFLAG,2       ; SET FLAG WORD FOR HEADER

                ;       OUTPUT HEADER

                LDA     0,@FWADR,2      ; GET HEADER POINTER WORD
                ISZ     FWADR,2 ; BUMP FORMAT POINTER
                LDA     3,FORM,2        ; GET ADDRESS OF FORMAT BLOCK
                LDA 1   T.SA,3          ; START OF FORMAT INFORMATION
                JSR     STRIO           ; OUTPUT HEADER

                ;       SEND HEADER TERMINATOR
```



```
                LDA     1,IFLAG,2       ; GET FLAG WORD
                LDA     0,.3    ; PICK UP MASK
                AND     1,0,SNR ; EXTRACT TERMINATION DISPLACEMENT
                JMP     HSPA    ; NO TERMINATOR, GO ON
                JSR     .+5     ; TABLE ADDRESS IN AC3
                0               ; NULL TERMINATOR
                72              ; "COLON"
                57              ; "SLASH"
                55              ; "DASH"
                ADD     0,3     ; TERM. CODE ADR. IN AC3
                LDA     1,0,3   ; GET CHARACTER
                LDA     0,OFLAG,2       ; SET OUTPUT FLAG WORD
                JSR     @ZAP,2  ; OUTPUT HEADER TERMINATION CHARACTER

                ;       SEND HEADER SPACES

HSPA:           LDA     1,IFLAG,2       ; GET FLAG WORD
                MOVR    1,1,SKP ; SHIFT FLAG WORD

BACK:           JMP     DOIT    ; CHAIN JUMP TO BEGINNING

                MOVR    1,1,SKP ; TWICE

.40:            40              ; ASCII "SPACE"

                LDA     0,.3    ; PICK UP MASK
                AND     0,1,SNR ; EXTRACT HEADER SPACE COUNT
                JMP     NEXT    ; NONE, GO ON
                STA     1,COUNT,2       ; STORE COUNTER
                LDA     0,OFLAG,2       ; SETUP OUTPUT FLAGS
                LDA     1,.40   ; LOAD A "SPACE"
                JSR     @ZAP,2  ; OUTPUT A "SPACE"
                DSZ     COUNT,2 ; DONE?
                JMP     .-2     ; NO, DO ANOTHER
                JMP     NEXT    ; SKIP AROUND LITERALS

                ;       FLAG-BIT MASKS

ELBIT:          BIT3            ; LINE ERASE FLAG
CRBIT:          BIT7            ; CR/LF TERMINATION
DTBIT:          BIT6            ; DATA FIELD TAGGED
LFBIT:          BIT0            ; LAST FIELD FLAG
HTBIT:          BIT5            ; HEADER FIELD BRIGHT
HBYTE:          BIT11           ; HIBYTE OUTPUT FLAG
LBYTE:          BIT12           ; LOBYTE OUTPUT FLAG
DIMIT:          BIT15           ; THIS FIELD TAGGED

                ;       SET UP FLAGS FOR DATA

NEXT:           LDA     1,OFLAG,2       ; GET OUTPUT FLAGS
                LDA     0,DIMIT ; GET TAG BIT
                COM     0,0,SKP ; INVERT MASK

.177:           177             ; 7 BIT MASK

                AND     0,1     ; FORCE TAG BIT LOW
                STA     1,OFLAG,2       ; RESET OUTPUT FLAGS FOR DATA
                LDA     1,IFLAG,2       ; GET INPUT FLAGS
```

```
00146'020763            LDA     0,DTBIT ; PICK UP MASK
00147'123405            AND     1,0,SNR ; TAG DATA?
00150'000405            JMP     .+5     ; NO, GO ON
00151'025065            LDA     1,OFLAG,2       ; PICK UP OUTPUT FLAGS
00152'020764            LDA     0,DIMIT ; PICK UP TAG BIT FLAG
00153'107000            ADD     0,1     ; SET TAG BIT
00154'045065            STA     1,OFLAG,2       ; RESET OUTPUT FLAGS FOR DATA

                ;       SEND DATA

00155'023062            LDA     0,@FWADR,2      ; GET DATA POINTER WORD
00156'011062            ISZ     FWADR,2 ; BUMP FORMAT POINTER
00157'025003$           LDA     1,DATA,2        ; GET ADDRESS OF DATA BLOCK
00160'004457            JSR     STRIO   ; OUTPUT DATA

                ;       SEND DATA TERM

00161'025064            LDA     1,IFLAG,2       ; GET FLAG WORD
00162'127121            ADDZL   1,1     ; MOVE DATA TERM. CODE
00163'127121            ADDZL   1,1     ; TO BITS 14 & 15
00164'125300            MOVS    1,1     ; OF AC1
00165'020663            LDA     0,.3    ; PICK UP MASK
00166'123405            AND     1,0,SNR ; EXTRACT TERMINATION DISPLACEMENT
00167'000412            JMP     DSPA    ; NO TERMINATOR, GO ON
00170'004405            JSR     .+5     ; TABLE ADDRESS IN AC3
00171'000000            0               ; NULL TERMINATOR
00172'000056            56              ; "DOT"
00173'000054            55              ; "COMMA"
00174'000055            55              ; "DASH"
00175'117000            ADD     0,3     ; TERM. CODE ADR. IN AC3
00176'025000            LDA     1,0,3   ; GET CHARACTER
00177'021065            LDA     0,OFLAG,2       ; SEND OUTPUT FLAG WORD
00200'007001$           JSR     @ZAP,2  ; OUTPUT DATA TERM. CHARACTER

                ;       SEND DATA SPACES

00201'025064    DSPA:   LDA     1,IFLAG,2       ; GET FLAG WORD
00202'127121            ADDZL   1,1     ; MOVE DATA SPACE COUNT
00203'125300            MOVS    1,1     ; TO BITS 14 & 15 OF AC1
00204'020641            LDA     0,.3    ; PICK UP MASK
00205'123405            AND     0,1,SNR ; EXTRACT DATA SPACE COUNT
00206'000407            JMP     OCRLF   ; NONE, GO ON
00207'045067            STA     1,COUNT,2       ; STORE COUNTER
00210'021065            LDA     0,OFLAG,2       ; SET UP OUTPUT FLAGS
00211'024703            LDA     1,.40   ; LOAD A "SPACE"
00212'007001$           JSR     @ZAP,2  ; OUTPUT A "SPACE"
00213'015067            DSZ     COUNT,2 ; DONE?
00214'000775            JMP     .-2     ; NO DO ANOTHER

                ;       SEND OPTIONAL CR/LF

00215'025064    OCRLF:  LDA     1,IFLAG,2       ; GET FLAG WORD
00216'027712            LDA     0,CRBIT ; PICK UP MASK
00217'123405            AND     1,0,SNR ; SHOULD IT BE DONE?
00220'000406            JMP     MAYBE   ; NO, MOVE ON
00221'021065            LDA     0,OFLAG,2       ; SET UP OUTPUT FLAGS
00222'024712            LDA     1,HBYTE ; GET HI BYTE FLAG BIT
00223'123000            ADD     1,0     ; SET UP TO OUTPUT BOTH BYTES
```



```
00224'024443            LDA     1,CRLF  ; GET CR/LF WORD
00225'007001$           JSR     @ZAP,2  ; OUTPUT CR/LF

                ;       TEST FOR DONE

00226'025064    MAYBE:  LDA     1,IFLAG,2       ; GET FLAG WORD
00227'020703            LDA     0,LFBIT ; GET DONE BIT MASK
00230'123404            AND     1,0,SZR ; WAS THIS LAST ELEMENT?
00231'000402            JMP     QUIT    ; IF IT WAS, RETURN
00232'000660            JMP     BACK    ; PROCESS NEXT FORMAT ELEMENT

00233'025002$   QUIT:   LDA 1   HOME,2  ; CURSOR REST POSITION
00234'102403            SUB     0,0
00235'007001$           JSR     @ZAP,2  ; REPOSITION
00236'003070            JMP     @RTRN,2 ; AND RETURN

                ;       STRING OUTPUT ROUTINE

00237'055071    STRIO:  STA     3,SRTN,2        ; SAVE RETURN ADDRESS
00240'034702            LDA     3,.177  ; PICK UP MASK
00241'117405            AND     0,3,SNR ; BYTE COUNTER IN AC3
00242'003071            JMP     @SRTN,2 ; NULL BYTE COUNT, RETURN
00243'055067            STA     3,COUNT,2       ; STORE COUNTER
00244'101300            MOVS    0,0     ; MOVE WORD ADDRESS TO LOWER BYTE
00245'115100            MOVL    0,3     ; SET CARRY TO "1ST BYTE" BIT
00246'101100            MOVL    0,0     ; BYTE ADDRESS IN LOWER 9 BITS
00247'034421            LDA     3,.777  ; PICK UP MASK
00250'117400            AND     0,3     ; EXTRACT BYTE ADDRESS
00251'125120            MOVZL   1,1     ; OFFSET BLOCK ADDRESS
00252'137000            ADD     1,3     ; BYTE ADDRESS OF DATA IN AC3
00253'055066            STA     3,POINT,2       ; STORE IT
00254'035066    LOOP:   LDA     3,POINT,2       ; FETCH BYTE ADDRESS OF DATA
00255'175220            MOVZR   3,3     ; CONVERT TO WORD ADDRESS
00256'025400            LDA     1,0,3   ; PICK UP DATA WORD
00257'125003            MOV     1,1,SNC ; WHICH BYTE?
00260'125300            MOVS    1,1     ; SWAP BYTES
00261'021065            LDA     0,OFLAG,2       ; SET UP OUTPUT FLAGS
00262'007001$           JSR     @ZAP,2  ; OUTPUT ONE BYTE
00263'011066            ISZ     POINT,2 ; POINT TO NEXT BYTE
00264'015067            DSZ     COUNT,2 ; DONE?
00265'000767            JMP     LOOP    ; NO, DO ANOTHER BYTE
00266'003071            JMP     @SRTN,2 ; YES, RETURN

00267'006412    CRLF:   6412            ; HI BYTE = CR, LO BYTE = LF
00270'000777    .777:   777             ; 9 BIT MASK

                .END
```

VAL	VALUE	DEF'N	REFERENCES						
	'112'	3:20	5:10						
		1:22	3:42						
		1:27	3:44						
		1:28	3:45						
		1:29	3:46						
		1:23	3:39						
		1:24	3:43						
		1:25	3:41						
		1:26	3:40						
		1:18	3:29	3:33	4:43	4:47	5:23	5:40	
		3:40	4:53						
	'267'	5:44	5:01						
	$X	1:09	4:13						
	'136'	3:46	2:44	3:51	4:05				
		1:49	3:24						
		4:37	4:24						
	'131'	3:41	4:01						
		1:37	1:34						
	'127'	3:39	2:09						
	$X	1:10	1:43	2:53					
		1:13	1:45	1:49	1:51	1:52	2:13	2:14	2:51
			2:52	4:11	4:12				
	'134'	3:44	4:57						
	$X	1:08	5:12						
	'119'	3:17	3:04						
	'133'	3:43	2:41						
	'64	1:15	1:54	2:17	2:40	3:01	3:17	3:58	4:18
			4:37	4:52	5:06				
	'13'	1:52	2:11						
	'135'	3:45	2:45						
	'132'	3:42	5:07						
	'254'	5:32	5:41						
	'63	1:14	1:47	2:01	2:06				
	'226'	5:06	4:55						
	'137'	3:50	3:23	3:35					
	'15'	4:52	4:42						
	'65	1:16	2:47	3:12	3:30	3:50	3:57	4:04	4:07
			4:32	4:44	4:56	5:37			
	'33'	2:13	2:03						
	'66	1:17	1:53	2:31	5:31	5:32	5:39		
	'233'	5:12	5:09						
	'70	1:31	1:39	5:15					
	'71	1:32	5:19	5:22	5:42				
	'61	1:12	1:38	2:07	2:33				
	'237'	5:19	2:55	4:14					
	$X	1:07	2:10	2:36	3:13	3:32	4:33	4:46	5:02
			5:14	5:38					
	'142'	3:54	2:15	5:20					
	'017'	1:57	1:46	1:54					
	'6'	2:24	2:18	3:02	3:26	4:22	4:40		
	'114'	3:24	3:31	4:45					
	'27'	5:45	5:27						

FLAGGED LINES: NONE

FORMAT EDIT (EDIT.)

The Format Edit routine is used to edit data blocks under format control. It is mainly used interactively, with input from a terminal which has a ticket displayed, but several routines make calls to it to use its features. Entry is made through one of the ten entry ports in the dispatch table, the first ten words of the program. Action taken at each entry port is as follows:

BACK: Moves the cursor position back one character. Will not backspace past the beginning of a field

TAB: Advances the cursor to the 1st character position of the next field which is marked as a tab stop field

REV: Works like TAB but in the reverse direction; moves to the beginning of a previous field.

ADV: Works like TAB but only stops on fields which are marked as advance stop fields.

TOP: Positions the cursor to the 1st character position of the 1st field.

TERM: Removes all data from the cursor position to the end of the field, leaving the cursor at the 1st character position of the next field.

DEL: Removes the data character at the cursor position and moves all following characters in the field left one position. The cursor does not move.

INS: Inserts a space at the cursor position and moves all following characters in the field right one position. The cursor does not move.

SPACE: Advances the cursor one character position. Will not space past the end of a field.

TEXT: Inserts a data character at the current cursor position and moves the cursor ahead one position.

```
                .HEAD   A I D S     F O R M A T     E D I T

                .TITLE  EDIT.

                ;       AIDS TABLE ASSIGNMENT

                .EXTD   .EDIT   ; EDIT VECTOR BLOCK ADDRESS
                .EXTD   .KILL   ; DELETE NON-RESIDENT CODE

                ;       TIT STORAGE WORD ASSIGNMENT

                .EXTD   ZAP     ; DISPLAY COMMAND ENTRY
                .EXTD   EDIT    ; EDIT BLOCK ADDRESS
                .EXTD   CHAR    ; INPUT DATA CHARACTER
                .EXTD   FORM    ; FORMAT BLOCK ADDRESS
                .EXTD   DATA    ; DATA BLOCK ADDRESS
                .EXTD   FIELD   ; CURRENT FIELD NUMBER
                .EXTD   INDEX   ; POSITION IN CURRENT FIELD
                .EXTD   END     ; END CONTROL WORD

      000060    BYCNT=  S0      ; BYTE COUNTER
      000061    POINT=  S1      ; BYTE POINTER
      000062    COUNT=  S2      ; LOOP COUNTER
      000063    RADAR=  S3      ; RECORD ADDRESS

      000071    TMP1=   T1
      000072    TMP2=   T2
      000073    SAVER=  T3      ; TEMPORARY RECORD ADDRESS
      000074    OFLAG=  T4      ; FLAGS FOR STRIO

      000053    RTRN=   R3      ; MAIN RETURN ADDRESS
      000070    SRET=   T0      ; SUBROUTINE RETURN ADDRESS

                .EXTD   MOD     ; DATA BLOCK UPDATE FLAG
                .EXTD   USE     ; DATA BLOCK USAGE COUNT
                .EXTD   REF     ; TICKET NUMBER IN BLOCK

                .ENT    TOP.,TAB.,TERM.,DATA.
                .END    EDIT.
                .NREL

                ;       DISPATCH TABLE

      000000    EDIT.:
      000000'000037'          BACK            ; BACKSPACE
      000001'000071'          TAB             ; TAB
      000002'000123'          REV             ; REVERSE TAB
      000003'000066'          ADV             ; ADVANCE STOP
      000004'000031'          TOP             ; TOP OF FORM
      000005'000163'          TERM            ; TERMINATE FIELD
      000006'000474'          DLETE           ; DELETE CHARACTER
      000007'000401'          INSRT           ; INSERT CHARACTER
      000010'000625'          SPACE           ; SPACE-OVER
      000011'000272'          TEXT            ; INPUT DATA

                ;       INITIALIZE ENTRY

      000012'024000     TOP.: SUB     0,0
```



```
      000013'041010$            STA 0   FIELD,2 ; ZERO FIELD NUMBER
      000014'055053     TAB.:   STA 3   RTRN,2  ; SAVE RETURN ADDRESS
      000015'035007$            LDA 3   DATA,2  ; DATA BLOCK ADDRESS
      000016'175014             MOV#    3,3 SZR ; IF DATA BLOCK,
      000017'035414$            LDA 3   USE,3   ; GET USE COUNT
      000020'175015             MOV#    3,3 SNR ; IF STILL ZERO,
      000021'000534             JMP     ERROR   ; SIGNAL ERROR
      000022'006[02]2$          JSR     @.KILL  ; DELETE CODE BLOCK

      000023'035006$            LDA 3   FORM,2  ; FORMAT BLOCK
      000024'025415$            LDA 1   REF,3   ; END CONTROL VECTOR
      000025'045012$            STA 1   END,2   ; SET END CONTROL
      000026'024001$            LDA 1   .EDIT   ; GET EDIT TABLE
      000027'045004$            STA 1   EDIT,2  ; INSTALL IN TIT
      000030'000404             JMP     TOP+3   ; LOOK FOR A STOP

      000031'055053     TOP:    STA 3   RTRN,2  ; SAVE RETURN
      000032'124400             SUB     0,0
      000033'041010$            STA 0   FIELD,2 ; ZERO FIELD NUMBER
      000034'020431             LDA 0   ASTOP   ; ADVANCE STOP BIT
      000035'041072             STA 0   TMP2,2  ; SAVE FOR TAB
      000036'000444             JMP     TIN     ; LOOK FOR A STOP

                        ;       BACKSPACE ROUTINE

      000037'055053     BACK:   STA 3   RTRN,2  ; SAVE RETURN ADDRESS
      000040'021011$            LDA 0   INDEX,2 ; GET POSITION IN FIELD
      000041'101005             MOV     0,0 SNR ; TEST FOR ZERO
      000042'000405             JMP     BIST    ; AT 1ST POSITION OF FIELD
      000043'015011$            DSZ     INDEX,2 ; DECREMENT POSITION IN FIELD
      000044'000401             NOP             ; (FOR DSZ)
      000045'004474     BUPDT:  JSR     MOVE    ; MOVE CURSOR
      000046'003053             JMP     @RTRN,2 ; RETURN

      000047'004502     BIST:   JSR     INITA   ; GET RECORD ADDRESS
      000050'025400             LDA 1   0,3     ; PICK UP FLAG WORD
      000051'020510             LDA 0   CONBT   ; FETCH CONTINUE BIT MASK
      000052'123405             AND     1,0 SNR ; CONTINUATION OF PREVIOUS FIELD?
      000053'000502             JMP     ERROR   ; NO, ERROR CONDITION
      000054'015010$            DSZ     FIELD,2 ; YES, MOVE TO PRECEDING FIELD
      000055'000401             NOP             ; (FOR DSZ)
      000056'025777             LDA 1   -1,3    ; GET DATA WORD FROM RECORD
      000057'034475             LDA 3   .177    ; FETCH MASK
      000060'167403             AND     3,1     ; GET LENGTH OF FIELD
      000061'124400             NEG     1,1     ; SUBTRACT ONE TO POINT TO ...
      000062'124000             COM     1,1     ; ... LAST POSITION IN FIELD
      000063'045011$            STA 1   INDEX,2 ; UPDATE POSITION IN FIELD
      000064'000761             JMP     BUPDT   ; UPDATE CURSOR

                        ;       ADVANCE STOP ROUTINE

      000065'040000     ASTOP:  40000   ; ADVANCE STOP BIT

      000066'020777     ADV:    LDA 0   ASTOP   ; PICK UP MASK
      000067'000403             JMP     TAB+1   ; LOOK FOR ADVANCE STOPS

                        ;       TAB STOP ROUTINE
```

```
                020000   TSTOP:  20000   ; TAB STOP BIT

                         TAB:    LDA 0   TSTOP   ; PICK UP MASK
                                 STA 3   RTRN,2  ; SAVE RETURN ADDRESS
                                 STA 0   TMP2,2  ; SAVE FOR LATER
                                 JSR     INITA   ; GET RECORD ADDRESS
                                 LDA 0   0,3     ; PICK UP FLAG WORD
                         TLOOP:  LDA 1   DONEB   ; FETCH MASK
                                 AND     0,1 SZR ; LAST FIELD?
                                 JMP     TLAST   ; YES
                                 ISZ     FIELD,2 ; MOVE TO NEXT FIELD
                         TIN:    JSR     INITA   ; GET RECORD ADDRESS
                                 LDA 0   0,3     ; PICK UP FLAG WORD
                                 LDA 1   TMP2,2  ; FETCH MASK
                                 AND     0,1 SNR ; IS THIS A STOP FIELD?
                                 JMP     TLOOP   ; NO, TEST FOR LAST FIELD

                         TOK:    SUB     0,0     ; CREATE A ZERO
                                 STA 0   INDEX,2 ; POINT TO 1ST POSITION IN FIELD
                                 JSR     MOVE    ; SET CURSOR POSITION
                                 JMP     @RTRN,2 ; RETURN

                         TLAST:  SUB     0,0     ; CREATE A ZERO
                                 STA 0   INDEX,2 ; POINT TO 1ST POSITION IN FIELD
                                 JSR     MOVE    ; SET CURSOR POSITION
                                 JMP     ERROR   ; INDICATE ERROR

                         ;       BACK TAB ROUTINE

                010000   RSTOP:  10000   ; REVERSE STOP BIT

                         REV:    LDA 0   RSTOP   ; PICK UP MASK
                                 STA 3   RTRN,2  ; SAVE RETURN ADDRESS
                                 STA 0   TMP2,2  ; SAVE FOR LATER
                                 LDA 0   INDEX,2 ; POSITION IN THIS FIELD
                                 MOV#    0,0 SZR ; IF NOT ZERO,
                                 JMP     BIN     ; TRY THIS FIELD

                         BLOOP:  LDA 0   FIELD,2 ; PICK UP FIELD NUMBER
                                 MOV#    0,0 SNR ; IF ZERO,
                                 JMP     TIN     ; SET TO TOP
                                 DSZ     FIELD,2 ; MOVE BACK ONE FIELD
                                 NOP             ; (FOR DSZ)

                         BIN:    JSR     INITA   ; GET RECORD ADDRESS
                                 LDA 0   0,3     ; FETCH FLAG WORD
                                 LDA 1   TMP2,2  ; FETCH MASK
                                 AND     0,1 SNR ; IS IT A STOP FIELD?
                                 JMP     BLOOP   ; NO, MOVE BACK ONE MORE
                         TOK1:   JMP     TOK     ; YES, STOP HERE

                         ;       ROUTINE TO REPOSITION THE CURSOR

                         MOVE:   STA 3   SRET,2  ; SAVE RETURN ADDRESS
                                 JSR     INITA   ; GET RECORD ADDRESS
                                 LDA 1   1,3     ; GET FIELD POSITION WORD
                                 LDA 0   INDEX,2 ; GET POSITION IN FIELD
                                 ADD     0,1     ; COMPUTE CURSOR POSITION
```



```
1   000146 020404           LDA 0   REPOS   ; UPDATE FLAG WORD
2   000147 035070           LDA 3   SRET,2  ; RETURN ADDRESS
3   000150 003003$          JMP     @ZAP,2  ; SET CURSOR POSITION
4
5   000151 000520   INITA:  JMP     INITB   ; LINK
6
7   000152 000100   REPOS:  100     ; POSITION UPDATE FLAG
8   000153 000007   .7:     7       ; ASCII BELL-CODE
9   000154 000177   .177:   177     ; SEVEN BIT MASK
10
11                  ;       ERROR ROUTINE
12
13  000155 035053   ERROR:  LDA 3   RTRN,2  ; RECOVER RETURN
14  000156 024775           LDA 1   .7      ; ASCII "BELL" CODE
15  000157 121400           INC     1,0     ; SET BASIC FLAGS
16  000160 003003$          JMP     @ZAP,2  ; EXIT THRU ZAP
17
18  000161 004000   CONBT:  4000    ; CONTINUATION FIELD FLAG
19  000162 100000   DONEB:  100000  ; LAST FIELD FLAG
20
21                  ;       FIELD TERMINATION ROUTINE
22
23  000163          TERM.:
24  000163 055053   TERM:   STA 3   RTRN,2  ; SAVE RETURN ADDRESS
25  000164 004505           JSR     INITB   ; GET RECORD ADDRESS
26  000165 055063           STA 3   RADAR,2 ; SAVE IT
27  000166 021401           LDA 0   1,3     ; GET POSITION OF DATA FIELD
28  000167 025011$          LDA 1   INDEX,2 ; GET POSITION IN DATA FIELD
29  000170 107000           ADD     0,1     ; GET ACTUAL PRESENT POSITION (CURSOR)
30  000171 102400           SUB     0,0     ; ZERO FLAG WORD FOR POSITION
31  000172 007003$          JSR     @ZAP,2  ; SET CURSOR POSITION
32  000173 035063           LDA 3   RADAR,2 ; RESTORE RECORD ADDRESS
33  000174 021403           LDA 0   3,3     ; GET DATA WORD
34  000175 024757           LDA 1   .177    ; PICK UP MASK
35  000176 123400           AND     1,0     ; EXTRACT DATA BYTE COUNT
36  000177 025011$          LDA 1   INDEX,2 ; GET DISTANCE INTO DATA
37  000200 122042           ADC0    1,0 SZC ; COMPUTE COUNTER VALUE
38  000201 000445           JMP     TDONE+1 ; PAST END OF FIELD
39  000202 101404           INC     0,0     ; INCREMENT
40  000203 041062           STA 0   COUNT,2 ; SAVE COUNT
41  000204 035063           LDA 3   RADAR,2 ; RESTORE RECORD ADDRESS TO AC3
42  000205 021403           LDA 0   3,3     ; PICK UP DATA WORD
43  000206 025007$          LDA 1   DATA,2  ; FETCH DATA BLOCK ADDRESS
44  000207 101300           MOVS    0,0     ; MOVE WORD ADDRESS TO LOWER BYTE
45  000210 115100           MOVL    0,3     ; SET CARRY TO "1ST BYTE" BIT
46  000211 101101           MOVL    0,0     ; BYTE ADDRESS IN LOWER 9 BITS
47  000212 034570           LDA 3   .777    ; PICK UP MASK
48  000213 117400           AND     0,3     ; EXTRACT BYTE ADDRESS
49  000214 021011$          LDA 0   INDEX,2 ; GET POSITION IN FIELD
50  000215 125120           MOVZL   1,1     ; OFFSET DATA BLOCK ADDRESS
51  000216 107000           ADD     0,1     ; MOVE PAST "OK" CHARACTERS
52  000217 137000           ADD     1,3     ; COMPUTE BYTE ADDRESS OF DATA
53  000220 055061           STA 3   POINT,2 ; SAVE IT
54  000221 035061   TRMLP:  LDA 3   POINT,2 ; PICK UP BYTE ADDRESS OF DATA
55  000222 175220           MOVZR   3,3     ; CONVERT TO WORD ADDRESS
56  000223 025400           LDA 1   0,3     ; PICK UP DATA WORD
57  000224 125003           MOV     1,1 SNC ; WHICH BYTE?
58  000225 125300           MOVS    1,1     ; SWAP BYTES
```

```
                00226'020553            LDA 0   HMASK   ; GET HI-BYTE MASK
                00227'137400            AND     0,1     ; REMOVE LOWER BYTE
                00230'020553            LDA 0   .40     ; PICK UP "SPACE" CHARACTER
                00231'137400            ADD     0,1     ; SET THIS BYTE TO AN ASCII "SPACE"
                00232'121403            MOV     1,0 SNC ; WERE BYTES SWAPPED?
                00233'101300            MOVS    0,0     ; SWAP THEM BACK AGAIN
                00234'041400            STA 0   0,3     ; RESET WORD IN DATA BLOCK
                00235'020541            LDA 0   FLAGS   ; PICK UP OUTPUT FLAG WORD
                00236'037443$           JSR     @ZAP,2  ; OUTPUT "SPACE"
                00237'011061            ISZ     POINT,2 ; POINT TO NEXT BYTE
                00240'035062            LDA 3   COUNT,2 ; PICK UP CHARACTER COUNT
                00241'175415            MOV#    3,3 SNR ; TEST FOR ZERO
                00242'000443            JMP     TDONE   ; DONE (ONLY HAD ONE)
                00243'155062            DSZ     COUNT,2 ; DONE?
                00244'000755            JMP     TRMLP   ; NO, DO ANOTHER BYTE
                00245'004524    TDONE:  JSR     MDFY    ; SET MODIFY FLAGS
                00246'035063            LDA 3   RADAR,2 ; PICK UP RECORD ADDRESS
                00247'025401            LDA 1   0,3     ; PICK UP FLAG WORD
                00250'020712            LDA 0   DONEB   ; FETCH TEST MASK
                00251'123404            AND     1,0 SZR ; LAST FIELD?
                00252'003053            JMP     @RTRN,2 ; YES, RETURN
                00253'011061$           ISZ     FIELD,2 ; POINT TO NEXT FIELD

                00254'025404            LDA 1   4,3     ; GET FLAGS FOR NEXT FIELD
                00255'020613            LDA 0   TSTOP   ; GET MASK
                00256'123414            AND#    1,0 SZR ; TEST FOR TAB STOP
                00257'000633            JMP     TOK     ; IS TAB STOP

                00260'034741            LDA 3   CONBT   ; PICK UP MASK
                00261'137435            AND     1,3 SNR ; CONTINUATION FIELD?
                00262'000611            JMP     TAB+2   ; NO, MOVE TO NEXT TAB STOP

                00263'124400            SUB     0,0     ; CREATE A ZERO
                00264'041011$           STA 0   INDEX,2 ; POINT TO FIRST CHARACTER
                00265'004654            JSR     MOVE    ; SET CURSOR POSITION
                00266'000675            JMP     TERM+1  ; DO IT AGAIN

                00267'000651    LINKE:  JMP     TOKI    ; LINKING JUMP
                00270'000651    MOVE2:  JMP     MOVE    ; LINKING JUMP
                000062          STASH=  COUNT   ; INPUT DATA STORAGE
                00271'000574    INITB:  JMP     INITC   ; LINK

                        ;       DATA CHARACTER INPUT ROUTINE

                00272'025005$   TEXT:   LDA 1   CHAR,2  ; PICK UP INPUT DATA
                00273'020574    DATA.:  LDA 0   .377    ; HALF-WORD MASK
                00274'137400            AND     0,1     ; EXTRACT DATA
                00275'045062            STA 1   STASH,2 ; SAVE FOR LATER
                00276'055053            STA 3   RTRN,2  ; SAVE RETURN ADDRESS
                00277'004565            JSR     INITC   ; GET RECORD ADDRESS
                00300'055063            STA 3   RADAR,2 ; SAVE IT
                00301'021403            LDA 0   3,3     ; FETCH DATA WORD
                00302'025402            LDA 1   .177    ; PICK UP MASK
                00303'137400            AND     0,1     ; COMPUTE FIELD LENGTH
                00304'021011$           LDA 0   INDEX,2 ; PICK UP POSITION IN FIELD
                00305'106042            ADCO    0,1 SZC ; PAST END OF FIELD?
                00306'000647    ERR:    JMP     ERROR   ; YES, ERROR
                00307'021403            LDA 0   3,3     ; PICK UP DATA WORD
```



```
                00310'025007$           LDA 1   DATA,2  ; GET DATA BLOCK ADDRESS
                00311'101300            MOVS    0,0     ; MOVE WORD ADDRESS TO LOWER BYTE
                00312'115100            MOVL    0,3     ; SET CARRY TO "1ST BYTE" BIT
                00313'101100            MOVL    0,0     ; BYTE ADDRESS IN LOWER 9 BITS
                00314'034466            LDA 3   .777    ; FETCH MASK
                00315'117400            AND     0,3     ; EXTRACT BYTE ADDRESS
                00316'021011$           LDA 0   INDEX,2 ; GET POSITION IN FIELD
                00317'125120            MOVZL   1,1     ; OFFSET DATA BLOCK ADDRESS
                00320'107000            ADD     0,1     ; MOVE PAST "OK" CHARACTERS
                00321'137224            ADDZR   1,3     ; COMPUTE WORD ADDRESS OF DATA
                00322'025401            LDA 1   0,3     ; PICK UP DATA WORD
                00323'125003            MOV     1,1 SNC ; WHICH BYTE?
                00324'125300            MOVS    1,1     ; SWAP BYTES
                00325'020454            LDA 0   HMASK   ; GET HI-BYTE MASK
                00326'123403            AND     1,0     ; REMOVE LOWER BYTE
                00327'025062            LDA 1   STASH,2 ; PICK UP DATA CHARACTER
                00330'123003            ADD     1,0 SNC ; SWAP AGAIN?
                00331'101300            MOVS    0,0     ; SWAP AGAIN!
                00332'041400            STA 0   0,3     ; INSTALL NEW DATA WORD
                00333'004436            JSR     MDFY    ; SET MODIFY FLAGS
                00334'004605            JSR     MOVE    ; SET CURSOR POSITION
                00335'035063            LDA 3   RADAR,2 ; PICK UP RECORD ADDRESS
                00336'035401            LDA 3   0,3     ; GET FLAG WORD
                00337'020532            LDA 0   DTBIT   ; "DATA TAGGED BIT" MASK
                00340'117404            AND     0,3 SZR ; TEST FOR TAG FLAG
                00341'034527            LDA 3   DIMIT   ; SET TAG FLAG
                00342'020434            LDA 0   FLAGS   ; GET BASIC FLAGS
                00343'163000            ADD     3,0     ; SET OUTPUT FLAGS
                00344'025062            LDA 1   STASH,2 ; PICK UP DATA CHARACTER
                00345'007003$           JSR     @ZAP,2  ; OUTPUT NEW CHARACTER
                00346'011011$   DMOV:   ISZ     INDEX,2 ; BUMP POSITION IN FIELD
                00347'004721            JSR     MOVE2   ; UPDATE CURSOR POSITION
                00350'035063            LDA 3   RADAR,2 ; FETCH RECORD ADDRESS
                00351'025403            LDA 1   3,3     ; GET DATA WORD
                00352'020602            LDA 0   .177    ; PICK UP MASK
                00353'123403            AND     1,0     ; GET FIELD LENGTH
                00354'025011$           LDA 1   INDEX,2 ; GET POSITION IN FIELD
                00355'122043            ADCO    1,0 SNC ; PAST FIELD?
                00356'003053            JMP     @RTRN,2 ; NO, RETURN
                00357'025403            LDA 1   0,3     ; PICK UP FLAG WORD
                00360'020602            LDA 0   DONER   ; FETCH TEST MASK
                00361'123404            AND     1,0 SZR ; LAST FIELD?
                00362'003053            JMP     @RTRN,2 ; YES, RETURN
                00363'025404            LDA 1   4,3     ; PICK UP FLAGS FOR NEXT FIELD
                00364'020413            LDA 0   CNBIT   ; PICK UP MASK
                00365'123405            AND     1,0 SNR ; TEST FOR CONTINUATION FIELD
                00366'003053            JMP     @RTRN,2 ; RETURN
                00367'011011$           ISZ     FIELD,2 ; MOVE TO NEXT FIELD
                00370'000677            JMP     LINKE   ; MOVE CURSOR

                00371'055071    MDFY:   STA 3   TMP1,2  ; SAVE RETURN ADDRESS
                00372'035007$           LDA 3   DATA,2  ; GET DATA BLOCK ADDRESS
                00373'102000            ADC     0,0     ; CREATE A -1
                00374'041413$           STA 0   MOD,3   ; SET "MODIFIED" FLAGS
                00375'003071            JMP     @TMP1,2 ; RETURN

                00376'000010    FLAGS:  10      ; BASIC OUTPUT FLAGS
                00377'004000    CNBIT:  4000    ; CONTINUATION FIELD FLAG
```

```
                        MOVE1:  JMP     MOVE2   ; LINKING JUMP
                        HMASK:  377*400 ; UPPER BYTE MASK
                        .777:   777     ; NINE BIT MASK
                        .40:    40      ; ASCII "SPACE"

                        ;       CHARACTER INSERTION CODE

                        INSRT:  STA 3   RTRN,2  ; SAVE RETURN ADDRESS
                                JSR     INITC   ; GET RECORD ADDRESS
                                STA 3   SAVER,2 ; SAVE IT FOR LATER USE
                                LDA 0   3,3     ; GET DATA WORD
                                LDA 3   MASK7   ; FETCH MASK
                                AND     0,3     ; EXTRACT LENGTH OF FIELD
                                STA 3   TMP1,2  ; SAVE IT FOR LATER
                                LDA 1   INDEX,2 ; GET POSITION IN FIELD
                                ADC0    1,3 SZC ; COMPUTE NUMBER OF CHARACTERS TO MOVE
                                JMP     @RTRN,2 ; ERROR IF PAST END OF FIELD
                                STA 3   COUNT,2 ; SET CHARACTER COUNTER
                                MOVS    0,0     ; MOVE WORD ADDRESS TO LOWER BYTE
                                MOVL    0,3     ; SET CARRY TO "1ST BYTE" BIT
                                MOVL    0,0     ; BYTE ADDRESS IN LOWER 9 BITS
                                LDA 3   .777    ; FETCH MASK
                                AND     0,3     ; EXTRACT BYTE ADDRESS
                                LDA 1   TMP1,2  ; PICK UP SIZE OF FIELD
                                ADD     1,3     ; MOVE TO END OF FIELD
                                LDA 1   DATA,2  ; GET ADDRESS OF DATA BLOCK
                                MOVZL   1,0     ; CONVERT TO BYTE ADDRESS
                                ADD     3,0     ; COMPUTE ABSOLUTE BYTE ADDRESS
                                LDA 1   .2      ; SUBTRACT 2 FROM BYTE ADDRESS ...
                                SUB     1,0     ; ... TO POINT TO LAST REMAINING CHARACTE
                                LDA 3   COUNT,2 ; PICK UP COUNT
                                MOV#    3,3 SNR ; TEST FOR ZERO
                                JMP     IONE    ; JUST NEED ONE "SPACE"
                        INCLP:  JSR     GET     ; PICK UP DATA CHARACTER
                                INC     0,0     ; POINT TO NEXT CHARACTER
                                JSR     PUT     ; REPLACE CHARACTER
                                LDA 1   .2      ; GET INCREMENT
                                SUB     1,0     ; MOVE BACK TWO CHARACTERS
                                DSZ     COUNT,2 ; DONE?
                                JMP     INCLP   ; NO, DO MORE
                        IONE:   INC     0,0     ; POINT TO INSERTED CHARACTER
                                LDA 1   .40     ; PICK UP AN ASCII "SPACE"
                                JSR     PUT     ; SET INSERTED CHARACTER TO A "SPACE"
                                LDA 3   SAVER,2 ; PICK UP RECORD ADDRESS
                                LDA 1   0,3     ; FETCH FLAG WORD
                                LDA 0   DTBIT   ; "DATA TAGGED BIT" MASK
                                AND     0,1 SZR ; TEST FOR TAG FLAG
                                LDA 1   DIMIT   ; SET TAG FLAG
                                LDA 0   FLAGS   ; GET BASIC FLAGS
                                ADD     1,0     ; SET OUTPUT FLAGS
                                STA 0   OFLAG,2 ; STORE FOR STRIO
                                LDA 0   3,3     ; PICK UP DATA WORD
                                JSR     STRIO   ; UPDATE DISPLAY
                                JSR     MOVE1   ; RESET CURSOR POSITION
                                JSR     MDFY    ; SET MODIFY FLAGS
                                JMP     @RTRN,2 ; RETURN

                        INITC:  JMP     INIT    ; LINK
```



```
00466'000002    .2:     2       ; NUMERIC CONSTANT
00467'000377    .377:   377     ; LOWER BYTE MASK
00470'000001    DIMIT:  1       ; TAGGED FIELD FLAG
00471'001000    DTBIT:  1000    ; TAGGED DATA FLAG
00472'000654    LINKG:  JMP     DMOV    ; LINKING JUMP
00473'000613    ERR1:   JMP     ERR     ; LINKING JUMP

                ;       CHARACTER DELETION CODE

00474'055053    DLETE:  STA 3   RTRN,2  ; SAVE RETURN ADDRESS
00475'004542            JSR     INIT    ; GET RECORD ADDRESS
00476'055073            STA 3   SAVER,2 ; SAVE IT FOR LATER USE
00477'021403            LDA 0   3,3     ; PICK UP DATA WORD
                        LDA 3   MASK7   ; FETCH MASK
                        AND     0,3     ; EXTRACT SIZE OF FIELD
00502'025011$           LDA 1   INDEX,2 ; GET POSITION IN FIELD
00503'136042            ADC0    1,3 SZC ; COMPUTE NUMBER OF CHARACTERS TO MOVE
00504'003053            JMP     @RTRN,2 ; ERROR IF PAST END OF FIELD
00505'055062            STA 3   COUNT,2 ; SET COUNTER
00506'101300            MOVS    0,0     ; MOVE WORD ADDRESS TO LOWER BYTE
00507'115100            MOVL    0,3     ; SET CARRY TO "1ST BYTE" BIT
00510'101100            MOVL    0,0     ; BYTE ADDRESS IN LOWER 9 BITS
00511'034427            LDA 3   MASK9   ; FETCH MASK
00512'117400            AND     0,3     ; EXTRACT BYTE ADDRESS
00513'025011$           LDA 1   INDEX,2 ; GET POSITION IN FIELD
00514'137000            ADD     1,3     ; MOVE PAST "OK" CHARACTERS
00515'025007$           LDA 1   DATA,2  ; FETCH DATA BLOCK ADDRESS
00516'125124            MOVZL   1,1     ; CONVERT TO BYTE ADDRESS
00517'167000            ADD     3,1     ; COMPUTE ABSOLUTE BYTE ADDRESS
00520'121400            INC     1,0     ; POINT TO 1ST CHARACTER TO BE MOVED
00521'035062            LDA 3   COUNT,2 ; PICK UP COUNT
00522'175015            MOV#    3,3 SNR ; TEST FOR ZERO
00523'000411            JMP     DWON    ; JUST NEED ONE "SPACE"
00524'004416    DECLP:  JSR     GET     ; PICK UP CHARACTER
00525'100400            NEG     0,0     ; MOVE BACK ONE ...
00526'100000            COM     0,0     ; ... CHARACTER POSITION
00527'004423            JSR     PUT     ; REPLACE CHARACTER
00530'024736            LDA 1   .2      ; PICK UP INCREMENT
00531'123001            ADD     1,0     ; MOVE FORWARD TWO CHARACTERS
00532'015062            DSZ     COUNT,2 ; DONE?
00533'000771            JMP     DECLP   ; NO, DO MORE
00534'100400    DWON:   NEG     0,0     ; MOVE TO LAST ...
00535'100001            COM     0,0     ; ... POSITION IN FIELD
00536'000710            JMP     IONE+1  ; OUTPUT "SPACE"

00537'000177    MASK7:  177     ; SEVEN BIT MASK
00540'001777    MASK9:  777     ; NINE BIT MASK
00541'177400    MASKH:  177400  ; HI BYTE MASK

                ;       BYTE MANIPULATION ROUTINES

00542'055070    GET:    STA 3   SRET,2  ; SAVE RETURN ADDRESS
00543'115224            MOVZR   0,3     ; CONVERT TO WORD ADDRESS
00544'025400            LDA 1   0,3     ; PICK UP DATA WORD
00545'125003            MOV     1,1 SNC ; WHICH BYTE?
00546'125300            MOVS    1,1     ; TOP ONE, SWAP BYTES
00547'034720            LDA 3   .377    ; FETCH MASK
00550'167400            AND     3,1     ; EXTRACT DATA BYTE
```

```
                        JMP     @SRET,2 ; RETURN

                PUT:    STA 3   SRET,2  ; SAVE RETURN ADDRESS
                        STA 1   TMP1,2  ; SAVE DATA BYTE
                        MOVZR   0,3     ; CONVERT TO WORD ADDRESS
                        LDA 0   0,3     ; PICK UP OLD DATA WORD
                        MOV     0,0 SNC ; WHICH BYTE?
                        MOVS    0,0     ; TOP, SWAP BYTES
                        LDA 1   MASKH   ; FETCH HI BYTE MASK
                        AND     1,0     ; REMOVE OLD BYTE
                        LDA 1   TMP1,2  ; PICK UP NEW DATA BYTE
                        ADD     0,1     ; INSERT NEW BYTE
                        MOV     1,1 SNC ; TOP BYTE?
                        MOVS    1,1     ; YES, SWAP BYTES
                        STA 1   0,3     ; STORE NEW DATA WORD
                        MOVL    3,0     ; RESET BYTE ADDRESS IN AC0
                        JMP     @SRET,2 ; RETURN

                ;       STRING OUTPUT ROUTINE

                STRIO:  STA 3   SRET,2  ; SAVE RETURN ADDRESS
                        LDA 3   MASK7   ; FETCH MASK
                        AND     0,3 SNR ; BYTE COUNTER IN AC3
                        JMP     @SRET,2 ; NULL BYTE COUNT, RETURN
                        LDA 1   INDEX,2 ; GET POSITION IN FIELD
                        SUB     1,3     ; DON'T COUNT "OK" CHARACTERS
                        STA 3   BYCNT,2 ; STORE COUNTER
                        MOVS    0,0     ; MOVE WORD ADDRESS TO LOWER BYTE
                        MOVL    0,3     ; SET CARRY TO "1ST BYTE" BIT
                        MOVL    0,0     ; BYTE ADDRESS IN LOWER 9 BITS
                        LDA 3   MASK9   ; FETCH MASK
                        AND     0,3     ; EXTRACT BYTE ADDRESS
                        ADD     1,3     ; MOVE PAST "OK" CHARACTERS
                        LDA 1   DATA,2  ; PICK UP DATA BLOCK ADDRESS
                        MOVZL   1,1     ; OFFSET BLOCK ADDRESS
                        ADD     1,3     ; COMPUTE ABSOLUTE BYTE ADDRESS
                        STA 3   POINT,2 ; STORE IT
                        LDA 0   OFLAG,2 ; PICK UP FLAG WORD
                LOOP:   LDA 3   POINT,2 ; FETCH BYTE ADDRESS OF DATA
                        MOVZR   3,3     ; CONVERT TO WORD ADDRESS
                        LDA 1   0,3     ; PICK UP DATA WORD
                        MOV     1,1 SNC ; TOP BYTE?
                        MOVS    1,1     ; YES, SWAP BYTES
                        JSR     @ZAP,2  ; OUTPUT ONE DATA BYTE
                        ISZ     POINT,2 ; POINT TO NEXT BYTE
                        DSZ     BYCNT,2 ; DONE?
                        JMP     LOOP    ; NO, DO ANOTHER BYTE
                        JMP     @SRET,2 ; YES, RETURN

                ;       SPACE OVER ROUTINE

                SPACE:  STA 3   RTRN,2  ; SAVE RETURN ADDRESS
                        JSR     INIT    ; GET RECORD ADDRESS
                        STA 3   RADAR,2 ; SAVE IT
                        LDA 0   3,3     ; FETCH DATA WORD
                        LDA 1   MASK7   ; PICK UP MASK
                        AND     0,1     ; COMPUTE FIELD LENGTH
                        LDA 0   INDEX,2 ; GET POSITION IN FIELD
```



```
1       ADCO    0,1 SZC ; PAST END OF FIELD?
2       JMP     ERR1    ; YES, ERROR
3       JMP     LINKG   ; MOVE CURSOR
4
5               ;       FORMAT ADDRESSING ROUTINE
6
7       INIT:   STA 3   TMP1,2  ; SAVE RETURN ADDRESS
8       LDA 1   FIELD,2 ; PICK UP FIELD NUMBER
9       ADDZL   1,1     ; *4 (FOUR WORD RECORDS)
10      LDA 3   FORM,2  ; GET FORMAT BLOCK ADDRESS
11      LDA 3   T.SA,3  ; GET START OF FORMAT
12      ADD     1,3     ; COMPUTE RECORD ADDRESS
13      JMP     @TMP1,2 ; RETURN
14
15              .END
```

SYMBOL	VALUE	DEF'N	REFERENCES						
	000066'	2:54	1:48						
OP	000065'	2:52	2:20	2:54					
T	000047'	2:35	2:29						
	000037'	2:26	1:45						
	000133'	3:45	3:37						
P	000126'	3:39	3:49						
	000135'	2:32	2:43						
IT	000060	1:21	9:27	9:46					
	$X	1:14	5:45						
T	000377'	6:58	6:45						
T	000161'	4:18	2:37	5:29					
IT	000062	1:23	4:40	5:11	5:14	5:40	7:18	7:31	7:39
			8:19	8:31	8:40				
	$X	1:16	2:03	4:43	6:01	6:52	7:26	8:27	9:34
	000273'	5:46	1:33						
	000524'	8:34	8:41						
IT	000474'	8:03	6:26	7:48					
	000174'	8:13	1:51						
	000346'	6:31	8:05						
	000162'	4:19	3:38	5:19	6:41				
	000171'	8:04	6:24	7:46					
	000534'	8:42	8:33						
IT	$X	1:13	2:14						
IT.	000064'	1:44	1:39						
	$X	1:19	2:12						
)	000306'	5:57	8:06						
	000273'	8:16	10:02						
	000155'	4:13	2:07	2:39	3:26	5:57			
LD	$X	1:17	2:01	2:19	2:40	3:11	3:39	3:42	5:22
			6:48	10:08					
	000376'	6:57	5:03	6:27	7:49				
	$X	1:15	2:10	10:10					
	000542'	8:52	7:34	8:34					
	000401'	7:02	5:01	6:14					
LP	000436'	7:34	7:40						
	$X	1:18	2:27	2:30	2:47	3:19	3:24	3:35	3:57
			4:28	4:36	4:49	5:34	5:55	6:07	6:31
			6:37	7:15	8:16	8:25	9:25	9:58	
	000637'	10:07	7:58	8:11	9:53				
A	000151'	4:05	2:35	3:06	3:12	3:45	3:55		
	000271'	5:41	4:05	4:25					
	000305'	7:58	5:41	5:50	7:09				
T	000404'	7:08	1:52						
	000445'	7:41	7:33	8:44					
	000267'	5:38	6:49						
	000472'	8:05	10:03						
	000613'	9:39	9:47						
7	000537'	8:46	7:12	8:14	9:22	9:56			
9	000543'	8:47	8:23	9:31					
1	000541'	8:48	9:09						
	371'	6:51	5:16	6:20	7:55				
	$X	1:34	6:54						
E	000141'	3:54	2:32	3:20	3:25	5:35	5:39	6:21	
1	000473'	7:01	7:54						
	000174'	5:39	6:32	7:01					

SYMBOL	VALUE	DEF'N	REFERENCES						
OFLAG	000074	1:29	7:51	9:38					
POINT	000061	1:22	4:53	4:54	5:10	9:37	9:39	9:45	
PUT	000552'	9:03	7:36	7:43	8:37				
RADAR	000063	1:24	4:26	4:32	4:41	5:17	5:51	6:22	6:33
			9:54						
REF	$X	1:36	2:11						
REPOS	000152'	4:07	4:01						
REV	000120'	3:32	1:47						
RSTOP	000117'	3:30	3:32						
RTRN	000053	1:31	2:02	2:17	2:26	2:33	3:04	3:21	3:33
			4:13	4:24	5:21	5:49	6:39	6:43	6:47
			7:08	7:17	7:56	8:10	8:18	9:52	
SAVER	000073	1:28	7:10	7:44	8:12				
SPACE	000625'	9:52	1:53						
SRET	000174	1:32	3:54	4:02	8:52	9:01	9:03	9:17	9:21
			9:24	9:48					
STASH	000062	5:40	5:48	6:16	6:29				
STRIO	000571'	9:21	7:53						
TAB	000071'	3:03	1:46	2:55	5:31				
TAB.	000014'	2:02	1:33						
TDONE	000245'	5:16	4:38	5:13					
TERM	000163'	4:24	1:50	5:36					
TERM.	000163'	4:23	1:38						
TEXT	000272'	5:45	1:54						
TIN	000102'	3:12	2:22	3:41					
TL	000113'	3:23	3:10						
TLOOP	000076'	3:08	3:16						
TMP1	000071	1:26	6:51	6:55	7:14	7:24	9:04	9:11	10:07
			10:13						
TMP2	000072	1:27	2:21	3:05	3:14	3:34	3:47		
TOK	000107'	3:18	3:50	5:27					
TOK1	000140'	3:50	5:38						
TOP	000031'	2:17	1:49	2:15					
TOP.	000012'	1:58	1:38						
TRMLP	000221'	4:54	5:15						
TSTOP	000070'	3:01	3:03	5:25					
USE	$X	1:35	2:05						
ZAP	$X	1:12	4:03	4:16	4:31	5:09	6:30	9:44	
.177	000154'	4:09	2:43	4:34	5:53	6:35			
.2	000466'	8:01	7:29	7:37	8:38				
.377	000467'	8:02	5:46	8:57					
.40	000403'	7:04	5:03	7:42					
./	000153'	4:08	4:14						
.777	000402'	7:03	4:47	6:05	7:22				
.EDIT	$X	1:07	2:13						
.KILL	$X	1:08	2:08						

FLAGGED LINES: NONE

UPDATE MONITOR (WATCH)

The update monitor is responsible for keeping the data blocks on disk up to date. Once a minute it writes the status table out on disk, and it continuously checks all resident data blocks to see if they have been modified or released. When it finds a data block whose modify flag is high, it updates the old copy on disk and sets the resident's modify flag low. When it finds a data block whose usage count is zero, it removes the block from the block list and releases the space it occupies in memory.


```
                        .HEAD   A I D S     U P D A T E     M O N I T O R
                        .TITLE  WATCH
                        .EXTD   SSB     ; SAVE STATUS BLOCK
                        .EXTD   LIST    ; DATA BLOCK LIST POINTER
                        .EXTD   LINK    ; GENERAL LINK DISPLACEMENT
                        .ENT    MOD,USE,BLK,REF
          000007    MOD=    7       ; DATA BLOCK UPDATE FLAG
          000006    USE=    6       ; DATA BLOCK USAGE COUNT
          000005    BLK=    5       ; BLOCK NUMBER IN TICKET
          000004    REF=    4       ; TICKET NUMBER IN BLOCK
                        .ENT    WATCH
                        .NREL
                        ; SAVE STATUS TABLE
00000'102520    SAVE:   SUBZL   0,0     ; DISK = 1
00001'034050            ENQ             ; WAIT FOR IT
00002'034041            LDA 3   DDOS    ; SYSTEM ADDRESS
00003'030475            LDA 2   NAME    ; FILE NAME
00004'024466            LDA 1   .SSB    ; BLOCK LIST
00005'020464            LDA 0   .4      ; BLOCK COUNT
00006'007403            JSR     @D.CYS,3; CALL OVERLAY
00007'000015            0.IMG           ; TO DUMP A FILE
00010'102523            SUBZL   0,0     ; DISK = 1
00011'034052            DEQ             ; GIVE IT UP
                        ; UPDATE TICKET DATA
00012'006040    WATCH:  IDLE            ; WAIT AWHILE
00013'020062            LDA 0   SECS    ; GET SECONDS COUNT
00014'101015            MOV#    0,0 SNR ; IF ZERO,
00015'000763            JMP     SAVE    ; SAVE STATUS TABLE
00016'030002$           LDA 2   LIST    ; DATA BLOCK LIST
00017'151015            MOV#    2,2 SNR ; IF EMPTY,
00020'000772            JMP     WATCH   ; WAIT SOME MORE
                        ; WAIT FOR THE DISK
00021'102520            SUBZL   0,0     ; DISK = 1
00022'034050            ENQ             ; GRAB IT
                        ; TEST UPDATE FLAG
00023'030002$   TRY:    LDA 2   LIST    ; NEXT DATA BLOCK
00024'021007            LDA 0   MOD,2   ; UPDATE FLAG
00025'101005            MOV     0,0 SNR ; IF ZERO,
00026'000413            JMP     OK1     ; OKAY
                        ; WRITE BLOCK TO DISK
00027'102400            SUB     0,0
00030'041007            STA 0   MOD,2   ; CLEAR UPDATE
```

```
                        LDA 3   .FWS    ; BUFFER OFFSET
                        ADD     3,2     ; BUFFER ADDRESS
                        LDA 0   .DA0,2  ; HIGH-ORDER ADDRESS
                        LDA 1   .DA1,2  ; LOW-ORDER ADDRESS
                        LDA 3   DDOS    ; DDOS SYSTEM TABLE
                        JSR     @D.WDS,3; CALL DISK WRITE
                        ABORT           ; ABORT ON ERROR
                        JMP     TRY     ; RE-TEST UPDATE

                ; TEST FOR NO USERS

                OK1:    LDA 2   LIST    ; DATA BLOCK
                        LDA 0   USE,2   ; USAGE COUNT
                        MOV     0,0 SZR ; IF ZERO,
                        JMP     OK2     ; OKAY

                ; SHORTEN BLOCK LIST

                        MOV     2,3 SKP ; DUPLICATE
                        MOV     1,3     ; BLOCK ADDRESS
                        LDA 1   LINK,3  ; NEXT DATA BLOCK
                        SEQ     1,2     ; IF NOT EQUAL,
                        JMP     .-3     ; KEEP LOOKING

                        LDA 2   LINK,2  ; NEXT DATA BLOCK
                        SNE     1,2     ; IF EQUAL,
                        SZR     2,2 SKP ; LIST EMPTY
                        STA 2   LINK,3  ; REPAIR LINKS
                        STA 2   LIST    ; AND BLOCK LIST

                ; DEALLOCATE SPACE

                        STA 1   BLOCK   ; SET ADDRESS
                        LDA 2   .IOCB   ; IOCB ADDRESS
                        .IO             ; DEALLOCATE
                        ABORT           ; WILL NEVER OCCUR
                        JMP     OUT

                ; MOVE TO NEXT BLOCK

                OK2:    LDA 2   LINK,2  ; NEXT BLOCK
                        STA 2   LIST    ; NOW CURRENT

                ; RELEASE THE DISK

                OUT:    SUBZL   0,0     ; DISK = 1
                        DEQ             ; GIVE IT UP

                        JMP     WATCH   ; GO WAIT AWHILE

        000001  .TXTM   1

                .4:     4
                .SSB:   SSB
                .FWS:   T.FW+B.FWS
        177775  .DA0=   B.DA0-B.FWS
        177775  .DA1=   B.DA1-B.FWS
```



```
 1  000074'000075' .IOCB:  .+1     ; DEALLOCATE IOC
 2  000075'000000          0
 3  000076'000001          1
 4  000077'000000  BLOCK:  0
 5
 6  001000'030101' NAME:   .+1     ; STATUS FILE
 7  001001'040511          .TXT    *AIDS.STATUS*
 8  001002'042123
 9  001003'027123
10  001004'052101
11  001005'052125
12  001006'051400
13
14         000062  SECS=   62      ; DINOS SECONDS WORD
15
16                 .END
```

SYMBOL	VALUE	DEF'N	REFERENCES				
[illegible]	000005	1:13	1:09				
[illegible]CK	000077'	3:04	2:33				
[illegible]X	$X	1:07	2:21	2:25	2:28	2:41	
[illegible]T	$X	1:06	1:38	1:49	2:12	2:29	2:42
[illegible]	000007	1:11	1:09	1:50	1:57		
[illegible]	[illegible]'	3:46	1:24				
[illegible]I	000041'	2:12	1:52				
[illegible]	[illegible]'	2:41	2:15				
[illegible]T	000066'	2:46	2:37				
[illegible]	000004	1:14	1:09				
[illegible]	[illegible]'	1:21	1:37				
[illegible]S	000062	3:14	1:35				
[illegible]	$X	1:05	2:54				
[illegible]	000023'	1:49	2:08				
[illegible]	000006	1:12	1:09	2:13			
[illegible]	[illegible]'	1:34	1:16	1:40	2:49		
[illegible]	000071'	2:53	1:26				
[illegible]O	177775	2:56	2:03				
[illegible]I	177776	2:57	2:04				
[illegible]S	000073'	2:55	2:01				
[illegible]B	000074'	3:01	2:34				
[illegible]	[illegible]'	2:54	1:25				

FLAGGED LINES: NONE

PRINTER MANAGER (PRINT)

The printer manager continuously monitors the printer flag. A printer flag high indicates that there is something in the printer file "XX.PRINTER" to be printed. When the flag is high, the printer monitor prints the contents of the print file and resets the print flag low.


```
                        .HEAD   A I D S     P R I N T E R     M A N A G E R
                        .TITLE  PRINT
177774          FILE=   4-T.FW  ; FILE ID FROM BUFFER
                ;       PRINTER TABLE OFFSETS
000001          DEPT=   1       ; DEPARTMENT
000010          FLAG=   10      ; PRINTER FLAGS
000011          TIT=    11      ; TABLE POINTER
                ;       TERMINAL TABLE OFFSETS
                .EXTD   ID      ; TERMINAL IDENTIFIER
                .EXTD   ZAP     ; PRINTER OUTPUT ENTRY
                .EXTD   SPOT    ; CARRIAGE POSITION
000051          RET=    R1      ; RETURN ADDRESS
000060          BLK=    S0      ; BLOCK ADDRESS
000061          BUFF=   S1      ; BUFFER ADDRESS
000062          OLD=    S2      ; OLD DATA POINTER
000063          NEW=    S3      ; NEW DATA POINTER
                .ENT    PRINT
                .NREL
00000'006040    PRINT:  IDLE            ; WAIT AWHILE
00001'021010            LDA 0   FLAG,2  ; PRINTER FLAGS
00002'101113            MOVL#   0,0 SNC ; IF FILE EMPTY,
00003'000775            JMP     PRINT   ; WAIT SOME MORE
                ; ALLOCATE BUFFER SPACE
00004'030562            LDA 2   .IOC1   ; CONTROL BLOCK
00005'006044            .IO             ; CALL ALLOCATE
00006'000772            JMP     PRINT   ; RETRY ON FAIL
00007'030002            LDA 2   2       ; PRINTER TABLE
00010'024561            LDA 1   BLK1    ; BLOCK ADDRESS
```

```
                ; WAIT FOR PRINTER IDLE

                        LDA 2   TIT,2   ; TERMINAL TABLE
                        LDA Ø   ID,2    ; TERMINAL NUMBER
                        ENQ             ; WAIT UNTIL FREE

                        STA 1   BLK,2   ; BLOCK ADDRESS
                        LDA Ø   .FWB    ; BUFFER OFFSET
                        ADD     Ø,1     ; BUFFER ADDRESS
                        STA 1   BUFF,2  ; SAVE FOR LATER

                        SUBZL   Ø,Ø     ; DISK = 1
                        ENQ             ; WAIT FOR IT

                ; SET UP FIRST SECTOR

                        LDA Ø   TOP     ; SET DATA START
                        STA Ø   OLD,2   ; PAST FILE NAME
                        LDA 2   2       ; RESTORE TABLE
                        LDA Ø   DEPT,2  ; GET DEPARTMENT
                        LDA 2   NAME    ; NAME POINTER
                        STA Ø   Ø,2     ; DEPT INTO NAME
                        LDA 3   DDOS    ; SYSTEM ADDRESS
                        JSR     @D.SRC,3; SEARCH FOR FILE
                        JMP     NONE    ; FILE NOT FOUND

                ; COMPUTE SECTOR ADDRESS

                        LDA 3   F.ID,2  ; FILE IDENTIFIER
                        LDA Ø   F.FST,2 ; HIGH-ORDER ADDRESS
                        LDA 1   F.BFP,2 ; LOW-ORDER ADDRESS
                        LDA 2   .377    ; HALF-WORD MASK
                        MOVZR   Ø,Ø     ; SHIFT HIGH-ORDER, THEN
                        ANDL    2,1     ; LOW-ORDER, WITH CARRY

                ; READ IN FIRST SECTOR

                        LDA 2   2       ; PRINTER TABLE
                        LDA 2   TIT,2   ; TERMINAL TABLE
                        LDA 2   BUFF,2  ; BUFFER ADDRESS
                        STA 3   FILE,2  ; SAVE IDENTIFIER
                        STA Ø   B.DAØ,2 ; HIGH-ORDER ADDRESS
                        STA 1   B.DA1,2 ; LOW-ORDER ADDRESS
                        JMP     READY   ; READ FIRST SECTOR
```



```
00050Ø006040 WAIT:   IDLE            ; WAIT A BIT

                ; READ IN A SECTOR

00051Ø102520 READ:   SUBZL   Ø,Ø     ; DISK = 1
00052Ø004050         ENQ             ; WAIT FOR IT
00053Ø021004 RETRY:  LDA Ø   B.DAØ,2 ; HIGH-ORDER ADDRESS
00054Ø025005         LDA 1   B.DA1,2 ; LOW-ORDER ADDRESS
00055Ø034041 READY:  LDA 3   DDOS    ; SYSTEM ADDRESS
00056Ø007405         JSR     @D.RED,3; READ A SECTOR
00057Ø000774         JMP     RETRY   ; RETRY ON ERROR
00060Ø102520         SUBZL   Ø,Ø     ; DISK = 1
00061Ø004052         DEQ             ; GIVE IT UP

                ; TEST END OF FILE

00062Ø021374         LDA Ø   FILE,2  ; FILE IDENTIFIER
00063Ø025012         LDA 1   B.ID,2  ; IDENTIFIER READ
00064Ø106414         SEQ     Ø,1     ; IF NOT EQUAL,
00065Ø000763         JMP     WAIT    ; WAIT AND RETRY

                ; COMPUTE NEXT SECTOR

00066Ø021007         LDA Ø   B.FOR,2 ; HIGH-ORDER ADDRESS
00067Ø025011         LDA 1   B.BFP,2 ; LOW-ORDER ADDRESS
00070Ø034515         LDA 3   .377    ; HALF-WORD MASK
00071Ø101220         MOVZR   Ø,Ø     ; SHIFT HIGH-ORDER, THEN
00072Ø167500         ANDL    3,1     ; LOW-ORDER, WITH CARRY
00073Ø100135         COMZL#  Ø,Ø SNR ; TEST FOR END OF FILE
00074Ø000413         JMP     LAST    ; THIS IS LAST SECTOR
00075Ø041004         STA Ø   B.DAØ,2 ; HIGH-ORDER ADDRESS
00076Ø045005         STA 1   B.DA1,2 ; LOW-ORDER ADDRESS

                ; PRINT THIS SECTOR

00077Ø030002         LDA 2   2       ; PRINTER TABLE
001ØØØ031011         LDA 2   TIT,2   ; TERMINAL TABLE
001Ø01024502         LDA 1   ALL     ; GET SECTOR END
001Ø02004444         JSR     DUMP    ; DUMP TO PRINTER
001Ø03176400         SUB     3,3     ; GET SECTOR TOP
001Ø04055062         STA 3   OLD,2   ; SET DATA START
001Ø05031061         LDA 2   BUFF,2  ; BUFFER ADDRESS
001Ø06000743         JMP     READ    ; GET NEXT SECTOR
```

```
                        ; PRINT LAST SECTOR

00107'030002  LAST:   LDA 2   2         ; PRINTER TABLE
00110'031011          LDA 2   TIT,2     ; TERMINAL TABLE
00111'021062          LDA 0   OLD,2     ; GET DATA START
00112'106112          SLT     0,1       ; IF NO MORE DATA,
00113'000414          JMP     QUIT      ; STOP PRINTING
00114'004432          JSR     DUMP      ; DUMP TO PRINTER
00115'031061  OVER:   LDA 2   BUFF,2    ; BUFFER ADDRESS
00116'000732          JMP     WAIT      ; WAIT AND RETRY

                        ; TURN OFF PRINTER

00117'020463  QUIT:   LDA 0   OFF       ; PRINTER-OFF
00120'025063          LDA 1   SPOT,2    ; POSITION WORD
00121'007002$         JSR     @ZAP,2    ; DISABLE PRINTER
00122'034002          LDA 3   2         ; PRINTER TABLE
00123'021410          LDA 0   FLAG,3    ; PRINTER FLAGS
00124'106212          MOVR#   0,0 SZC   ; IF FILE BUSY,
00125'000771          JMP     OVER      ; WAIT AND RETRY

                        ; CLEAN UP AFTER

00126'021001$ DONE:   LDA 0   ID,2      ; TERMINAL NUMBER
00127'004052          DEQ               ; RELEASE IT
00130'025061          LDA 1   BLK,2     ; BUFFER ADDRESS
00131'044446          STA 1   BLK2      ; SET INTO IOC
00132'024442          LDA 2   .IOC2     ; CONTROL BLOCK
00133'006044          .IO               ; DEALLOCATE
00134'006047          ABORT             ; ALWAYS SKIPS
00135'034002          LDA 2   2         ; PRINTER TABLE
00136'102400          SUB     0,0
00137'041414          STA 0   FLAG,2    ; SET FILE EMPTY
00140'000641          JMP     PRINT     ; GO WAIT AWHILE

                        ; FILE NOT FOUND

00141'102521  NONE:   SUBZL   0,0       ; DISK = 1
00142'004052          DEQ               ; GIVE IT UP
00143'034002          LDA 2   2         ; RESTORE TABLE
00144'031011          LDA 2   TIT,2     ; RECOVER TIT
00145'000761          JMP     DONE      ; AND QUIT
```



```
                        ; DUMP TEXT TO PRINTER

00146'055051  DUMP:   STA 3   RET,2     ; SAVE RETURN
00147'045063          STA 1   NEW,2     ; TEXT LIMIT

00150'025062  LOOP:   LDA 1   OLD,2     ; TEXT START
00151'021063          LDA 0   NEW,2     ; TEXT LIMIT
00152'106513          SGT     0,1       ; IF NO TEXT
00153'003051          JMP     @RET,2    ; RETURN

00154'020424          LDA 0   C1        ; FOR LO-BYTE
00155'125223          MOVZR   1,1 SNC   ; WORD INDEX
00156'020423          LDA 0   C2        ; FOR HI-BYTE

00157'035061          LDA 3   BUFF,2    ; BUFFER ADDRESS
00160'137000          ADD     1,3       ; ADD WORD OFFSET
00161'025415          LDA 1   B.FWD,3   ; FETCH DATA WORD

00162'007002$         JSR     @ZAP,2    ; PRINT ONE BYTE
00163'011062          ISZ     OLD,2     ; ADVANCE INDEX
00164'000764          JMP     LOOP      ; AND REPEAT
00165'003051          JMP     @RET,2    ; RETURN

00166'000167' .IOC1:  .+1       ; IOC TO ALLOCATE
00167'000000          0
00170'000000          0
00171'000000  BLK1:   0
00172'000000          0
00173'000420          T.FW+B.LEN+1

00174'000175' .IOC2:  .+1       ; IOC TO DEALLOCATE
00175'000000          0
00176'000001          1
00177'000000  BLK2:   0

00200'000010  C1:     10
00201'000020  C2:     20
00202'004 1000  OFF:    40000
      000201' TOP=    C2
00203'000764  ALL:    B.LEN-B.FWD*2
00204'000010  .FWB:   T.FW
00205'000377  .377:   377

00206'000247' NAME:   .+1
      000001  .TXTM   1
00207'054130  .TXT    *XX.PRINTER*
00210'027120
00211'051111
00212'047124
00213'042522
00214'000000

                      .END
```

SYMBOL	VALUE	DEF'N	REFERENCES				
	000203'	5:40	3:38				
	000063	1:21	2:07	4:26			
A1	000171'	5:27	1:40				
A2	000177'	5:34	4:27				
AF	000361	1:22	2:10	2:40	3:42	4:09	5:15
	000200'	5:36	5:11				
	000201'	5:37	5:13	5:39			
	000001	1:09	2:21				
	000126'	4:24	4:42				
	000146'	5:03	3:39	4:08			
BLE	177774	1:05	2:41	3:17			
BS	000010	1:10	1:30	4:18	4:33		
	$X	1:15	2:14	4:24			
	000107'	4:03	3:33				
	000154'	5:26	5:21				
CE	000206'	5:44	2:21				
	000063	1:24	5:04	5:07			
	000141'	4:38	2:25				
	000202'	5:38	4:14				
	000042	1:23	2:18	3:41	4:05	5:06	5:20
CR	000115'	4:09	4:23				
DIT	000003'	1:29	1:26	1:32	1:38	4:34	
DIT	000117'	4:14	4:07				
EAD	000051'	3:06	3:43				
ERY	000055'	3:09	2:44				
)	000051	1:19	5:03	5:09	5:22		
/	000053'	3:07	3:11				
ST	$X	1:17	4:15				
IT	000011	1:11	2:03	2:39	3:37	4:04	4:41
P	000221'	5:39	2:17				
IT	000050'	3:01	3:20	4:10			
>	$X	1:16	4:16	5:19			
?	000206'	5:42	2:32	3:26			
@	000204'	5:41	2:08				
LOC	000166'	5:24	1:36				
LOC2	000174'	5:31	4:28				

FLAGGED LINES: NONE

KEYBOARD DATA ENTRY (AWAIT)

The keyboard data entry routine reads keyboard input using ZIP (keyboard input routine). It dispatches function requests to the applicable function routine and sends data on through the EDIT vector in the terminal information table.


```
                        .HEAD   A I D S     K E Y B O A R D     M O N I T O R
                        .TITLE  AWAIT
                        .EXTD   ID
                        .EXTD   ZIP,ZAP
                        .EXTD   EDIT,CHAR
                        .ENT    AWAIT
                        .NREL
00000'007002$ AWAIT:    JSR     @ZIP,2  ; GET INPUT
00001'021001$           LDA     0,ID,2  ; TERMINAL ID
00002'004050            ENQ             ; WAIT FOR IT
00003'045005$           STA     1,CHAR,2; SAVE INPUT
00004'135000            MOV     1,3     ; DUPLICATE
00005'020417            LDA     0,.30   ; TEST FOR
00006'106423            SUBZ    0,1,SNC ; FUNCTION
00007'000406            JMP     FCN     ; IF SO, GO
00010'020413            LDA     0,.11   ; TEST FOR
00011'122113            SGE     1,0     ; DATA VALUE
00012'121000            MOV     1,0     ; DATA -> 11
00013'035004$           LDA     3,EDIT,2; EDIT TABLE
00014'175015            MOV#    3,3 SNR ; IF NO EDIT,
00015'020410 FCN:       LDA     0,.SYS  ; SYSTEM TABLE
00016'117000            ADD     0,3     ; COMPUTE ENTRY
00017'007400            JSR     @0,3    ; AND DISPATCH
00020'021001$           LDA     0,ID,2  ; TERMINAL ID
00021'004052            DEQ             ; RELINQUISH
00022'000756            JMP     AWAIT   ; AND REPEAT
00023'000011 .11:       11
00024'000030 .30:       30
00025'000026' .SYS:     SYS             ; SYSTEM TABLE
```

```
                ; FUNCTION DISPATCH TABLE

                .EXTN   MAKE.,FIND.
                .EXTN   UNIT.,STAT.
                .EXTN   CALL.,LEAD.
                .EXTN   WANT.,NEED.
                .EXTN   FILE.,COPY.
                .EXTN   MEMO.,ZERO.
                .EXTN   TOP.,END.
                .EXTN   TEST.

                SYS:    ERROR           ; 0 = IGNORE
                        CALL.           ; 1 = INVOKE
                        MAKE.           ; 2 = EVENT
                        UNIT.           ; 3 = UNIT
                        FIND.           ; 4 = TICKET
                        STAT.           ; 5 = STATUS
                        WANT.           ; 6 = WANT RADIO
                        NEED.           ; 7 = NEED RADIO

                        LEAD.           ; 10 = LEADS
                        NONE            ; 11 = SEND
                        COPY.           ; 12 = PRINT
                        NONE            ; 13 = BLANK
                        TEST.           ; 14 = BLANK
                        FILE.           ; 15 = FILE
                        MEMO.           ; 16 = MEMO
                        ZERO.           ; 17 = CLEAR

                        TEST.           ; 20 = DEBUG
                        NONE            ; 21
                        NONE            ; 22
                        NONE            ; 23
                        NONE            ; 24
                        NONE            ; 25
                        TOP.            ; 26 = TOP
                        END.            ; 27 = END

                ; UNRECOGNIZED ENTRY

                        7
                ERROR:  LDA 1   .-1     ; ASCII BELL-CODE
                        INC     1,0     ; ONE CHAR OUTPUT
                        JMP     @ZAP,2  ; EXIT THRU ZAP

                ; UNIMPLEMENTED FUNCTION

                NONE:   JMP     0,3     ; IMMEDIATE RETURN

                .END
```


SYMBOL	VALUE	DEF'N	REFERENCES						
AWAIT	000000'	1:12	1:09	1:34					
CALL.	'X	2:05	2:13						
CHAR	$X	1:07	1:16						
COPY.	'X	2:07	2:23						
EDIT	$X	1:07	1:26						
END.	'X	2:09	2:37						
ERROR	000057'	2:42	2:12						
FCN	000015'	1:28	1:24						
FILE.	'X	2:07	2:26						
FIND.	'X	2:03	2:16						
ID	$X	1:05	1:13	1:32					
LEAD.	'X	2:05	2:21						
MAKE.	'X	2:03	2:14						
MEMO.	'X	2:08	2:27						
NEED.	'X	2:06	2:19						
NONE	000062'	2:48	2:22	2:24	2:31	2:32	2:33	2:34	2:35
STAT.	'X	2:04	2:17						
SYS	000026'	2:12	1:38						
TEST.	'X	2:10	2:25	2:30					
TOP.	'X	2:09	2:36						
UNIT.	'X	2:04	2:15						
WAIT.	'X	2:06	2:18						
ZAP	$X	1:06	2:44						
ZERO.	'X	2:08	2:28						
ZIP	$X	1:06	1:12						
.1	000023'	1:36	1:22						
.30	000024'	1:37	1:18						
.SYS	000025'	1:38	1:28						

FLAGGED LINES: NONE

INVOKE FUNCTION (CALL.)

The Invoke Function is used to run utility programs. The routine types a colon on the terminal output device and awaits keyboard input of a file name. When the input string is terminated, with the tab or return keys, the invoke function appends a dot to the input filename and tries to run the named program. If the named program is not present, the routine types a question mark and rings the bell.

Some routines pass a file name pointer to the invoke function to initiate a process, bypassing keyboard input.


```
                .HEAD   A I D S    I N V O K E    F U N C T I O N

                .TITLE  CALL.

                .EXTD   .KILL   ; DELETE CODE BLOCK
                .EXTD   TAD     ; TAG-LINE ADDRESS

                .EXTD   ZAP     ; DISPLAY COMMAND ENTRY
                .EXTD   EDIT    ; EDIT COMMAND ENTRY
                .EXTD   CHAR    ; INPUT DATA CHARACTER
                .EXTD   HOME    ; CURRENT CURSOR POSITION
                .EXTD   FLAG    ; TERMINAL FLAG WORD
                .EXTD   RUN     ; NON-RESIDENT CODE
                .EXTD   END     ; END CONTROL WORD

                ;       TIT STORAGE WORD ASSIGNMENT

        000050  RTRN=   R0      ; MAIN RETURN ADDRESS

        000067  POINT=  S7      ; DATA BYTE POINTER
        000066  COUNT=  S6      ; CHARACTER COUNTER

        000100  FN=     100     ; FILE NAME OFFSET
        000110  CB=     110     ; I/O CONTROL BLOCK

                        .ENT    CALL.,OLAY.,KILL.
                        .NREL

00000'055050    CALL.:  STA 3   RTRN,2  ; SAVE RETURN ADDRESS
00001'006001$           JSR     @.KILL  ; DELETE CODE BLOCK
00002'034533            LDA 3   .FN     ; FILE NAME OFFSET
00003'157000            ADD     2,3     ; WORD ADDRESS OF FILE NAME
00004'161120            MOVZL   3,0     ; BYTE ADDRESS OF FILE NAME
00005'041067            STA 0   POINT,2 ; INITIALIZE BYTE POINTER

00006'024432            LDA 1   .10     ; EIGHT WORDS FOR NAME
00007'045066            STA 1   COUNT,2 ; SET UP COUNTER
00010'126400            SUB     1,1     ; MAKE A ZERO
00011'045400            STA 1   0,3     ; CLEAR A WORD
00012'175400            INC     3,3     ; BUMP POINTER
00013'015066            DSZ     COUNT,2 ; DONE?
00014'000775            JMP     .-3     ; NO, CLEAR ANOTHER

00015'034430            LDA 3   TABLE   ; GET ADDRESS OF EDIT TABLE
00016'055004$           STA 3   EDIT,2  ; SET UP FOR DISPATCH ENTRY
00017'034001$           LDA 3   .KILL   ; ABRUPT EXIT ENTRY
00020'055011$           STA 3   END,2   ; SET END CONTROL

00021'021007$           LDA 0   FLAG,2  ; GET DEVICE FLAGS
00022'024422            LDA 1   MASKT   ; FETCH MASK
00023'107405            AND     0,1 SNR ; POSITIONING NECESSARY?
00024'000405            JMP     PCR     ; NO, JUST A LINE FEED

00025'024002$           LDA 1   TAD     ; GET STARTING POSITION
00026'020511            LDA 0   E0      ; FLAGS FOR POSITION
00027'007003$           JSR     @ZAP,2  ; POSITION
00030'000404            JMP     PCS     ; CONTINUE
```

```
00031'020512  PCR:    LDA 0   D2      ; GET FLAG WORD
00032'024413          LDA 1   CRLF    ; GET ASCII <15><12>
00033'007003$         JSR     @ZAP,2  ; ADVANCE
00034'020515  PCS:    LDA 0   E2      ; GET FLAG WORD
00035'024436          LDA 1   COLON   ; ASCII ": "
00036'007003$         JSR     @ZAP,2  ; SEND ": "
00037'003050          JMP     @RTRN,2 ; RETURN

00040'000011  .10:    10
00041'000056  DOT:    ".
00042'036412  CRLF:   15*400+12
00043'035440  COLON:  ":*400+40
00044'100000  MASKT:  100000

00045'000045' TABLE:  .+1

00046'000065'         BACK            ; BACKSPACE
00047'000076'         TAB             ; TAB
00050'000103'         REV             ; REVERSE TAB
00051'000061'         BELL            ; ADVANCE
00052'000061'         BELL            ; TOP
00053'000121'         RET             ; RETURN
00054'000061'         BELL            ; INSERT CHARACTER
00055'000061'         BELL            ; DELETE CHARACTER
00056'000116'         SPACE           ; SPACE
00057'000144'         DATA            ; DATA ENTRY

00060'000007          7
00061'024777  BELL:   LDA 1   .-1     ; ASCII "BELL"
00062'020466          LDA 0   DI      ; SET FLAGS
00063'007003$         JSR     @ZAP,2  ; RING BELL
00064'003050          JMP     @RTRN,2 ; RETURN

00065'055061  BACK:   STA 3   RTRN,2  ; SAVE RETURN
00066'021066          LDA 0   COUNT,2 ; WHERE ARE WE?
00067'101005          NEG     0,0 SNR ; BEGINNING?
00070'000771          JMP     BELL    ; YES, RING IT

00071'110003          COM     0,0
00072'041066          STA 0   COUNT,2 ; DECREMENT COUNTER
00073'015006$         DSZ     HOME,2  ; AND CURRENT POSITION
00074'004503          JSR     MOVE    ; MOVE CURSOR
00075'003050          JMP     @RTRN,2 ; AND RETURN

00076'055051  TAB:    STA 3   RTRN,2  ; SAVE RETURN
00077'004503          JSR     GET     ; GET THIS CHARACTER
00100'125005          MOV     1,1 SNR ; NULL?
00101'000421          JMP     RET+1   ; YES, TERMINATE

00102'011060          ISZ     COUNT,2 ; BUMP COUNTER
00103'011056$         ISZ     HOME,2  ; AND POSITION
00104'000773          JMP     TAB+1   ; AND LOOK AGAIN

00105'055051  REV:    STA 3   RTRN,2  ; SAVE RETURN
00106'021066          LDA 0   COUNT,2 ; GET COUNTER
00107'101005          MOV     0,0 SNR ; IS IT ZERO?
00110'000751          JMP     BELL    ; YES, LET 'EM KNOW
```



```
00111'015006$         DSZ     HOME,2  ; DECREMENT POSITION
00112'015066          DSZ     COUNT,2 ; AND COUNTER. DONE?
00113'000776          JMP     .-2     ; NO, PROCEED BACKWARD
00114'004463          JSR     MOVE    ; MOVE CURSOR
00115'003050          JMP     @RTRN,2 ; AND RETURN

00116'021066  SPACE:  LDA 0   COUNT,2 ; GET COUNTER
00117'101015          MOV#    0,0 SNR ; IF ZERO,
00120'001400          JMP     0,3     ; IGNORE

00121'055050  RET:    STA 3   RTRN,2  ; SAVE RETURN
00122'021066          LDA 0   COUNT,2 ; GET COUNTER
00123'101015          MOV#    0,0 SNR ; IF ZERO,
00124'000571          JMP     RAD     ; FORGET IT
00125'024714          LDA 1   DOT     ; GET ASCII "."

00126'004466  REP:    JSR     PUT     ; APPEND TO PROGRAM NAME
00127'011066          ISZ     COUNT,2 ; BUMP COUNTER
00130'004452          JSR     GET     ; GET THIS CHARACTER
00131'125005          MOV     1,1 SNR ; NULL?
00132'000502          JMP     TERM    ; YES, TERMINATE
00133'126400          SUB     1,1     ; CREATE A NULL
00134'000772          JMP     REP     ; AND CONTINUE

00135'000100  .FN:    FN
00136'000110  .CB:    CB
00137'010100  E0:     10100   ; POSITION, ERASE
00140'010110  E1:     10110   ; TYPE ONE, ERASE
00141'010130  E2:     10130   ; TYPE TWO, ERASE
00142'000110  D1:     110     ; TYPE ONE
00143'000130  D2:     130     ; TYPE TWO

00144'021066  DATA:   LDA 0   COUNT,2 ; GET COUNTER
00145'024571          LDA 1   .14     ; LIMIT
00146'106112          SLT     0,1     ; PAST PROGRAM NAME?
00147'000712          JMP     BELL    ; YES, COMPLAIN

00150'025005$         LDA 1   CHAR,2  ; GET DATA CHARACTER
00151'020563          LDA 0   .60     ; LOWEST ALLOWED
00152'122113          SGE     1,0     ; IN RANGE?
00153'000706          JMP     BELL    ; NO, TOO LOW (<60)

00154'020554          LDA 0   .137    ; EDGE OF LOWER CASE
00155'122513          SGT     1,0     ; LOWER CASE?
00156'000403          JMP     .+3     ; NO, GO ON
00157'020556          LDA 0   .40     ; MAYBE, GET DISPLACEMENT
00160'106400          SUB     0,1     ; CONVERT TO UPPER CASE

00161'020550          LDA 0   .132    ; HIGHEST ALLOWED
00162'122512          SLE     1,0     ; IN RANGE?
00163'000676          JMP     BELL    ; NO, TOO HIGH (>132)

00164'020547          LDA 0   .71     ; HIGHEST DIGIT
00165'122513          SGT     1,0     ; ABOVE DIGITS?
00166'000404          JMP     ROK     ; NO, GO ON
00167'020543          LDA 0   .100    ; EDGE OF UPPER CASE
00170'122513          SGT     1,0     ; BELOW LETTERS?
00171'000670          JMP     BELL    ; YES, INVALID
```

```
                                
00172 020750  ROK:    LDA 0  D1       ; SET FLAGS
00173 007003$         JSR    @ZAP,2   ; DISPLAY DATA CHARACTER
00174 004420          JSR    PUT      ; INSERT CHARACTER INTO FILE NAME
00175 011065          ISZ    COUNT,2  ; BUMP COUNTER
00176 003050          JMP    @RTRN,2  ; AND RETURN
                                
00177 102400  MOVE:   SUB    0,0      ; SET FLAGS TO POSITION
00200 025006$         LDA 1  HOME,2   ; GET NEW POSITION
00201 003003$         JMP    @ZAP,2   ; MOVE CURSOR
                                
00202 054520  GET:    STA 3  SRTN     ; SAVE RETURN
00203 035067          LDA 3  POINT,2  ; GET BYTE POINTER
00204 021066          LDA 0  COUNT,2  ; AND COUNTER
00205 117220          ADDZR  0,3      ; MAKE WORD ADDRESS
00206 025000          LDA 1  0,3      ; PICK UP DATA WORD
00207 125003          MOV    1,1 SNC  ; WHICH BYTE?
00210 125300          MOVS   1,1      ; TOP ONE, SWAP BYTES
00211 034514          LDA 3  MASKL    ; FETCH MASK
00212 167400          AND    3,1      ; EXTRACT DATA BYTE
00213 002507          JMP    @SRTN    ; RETURN
                                
00214 054506  PUT:    STA 3  SRTN     ; SAVE RETURN
00215 044506          STA 1  TEMP     ; SAVE DATA BYTE
00216 035067          LDA 3  POINT,2  ; GET BYTE POINTER
00217 021066          LDA 0  COUNT,2  ; AND COUNTER
00220 117220          ADDZR  0,3      ; MAKE WORD ADDRESS
00221 021400          LDA 0  0,3      ; PICK UP OLD DATA WORD
00222 101003          MOV    0,0 SNC  ; WHICH BYTE?
00223 101300          MOVS   0,0      ; TOP, SWAP BYTES
00224 025000          LDA 1  MASKH    ; FETCH HI-BYTE MASK
00225 123400          AND    1,0      ; REMOVE OLD CYTE
00226 024475          LDA 1  TEMP     ; PICK UP NEW BYTE
00227 107000          ADD    0,1      ; INSERT NEW BYTE
00230 125003          MOV    1,1 SNC  ; TOP BYTE?
00231 125300          MOVS   1,1      ; YES, SWAP DEM BYTES!
00232 045400          STA 1  0,3      ; STORE NEW DATA WORD
00233 002467          JMP    @SRTN    ; RETURN
                                
00234 020703  TERM:   LDA 0  E0       ; GET ERASE FLAGS
00235 004743          JSR    MOVE+1   ; CLEAR REST OF LINE
00236 024477          LDA 1  .FN      ; INDEX TO NAME STORAGE
00237 147001          ADD    2,1 SKP  ; POINT TO NAME
                                
00240 025000  OLAY.:  LDA 1  0,3      ; NAME POINTER
00241 044462          STA 1  TEMP     ; SAVE FOR LATER
00242 004421          JSR    KILL.    ; DELETE OLD CODE
00243 034673          LDA 3  .CB      ; INDEX TO IOCB
00244 157000          ADD    2,3      ; POINT TO IOCB
00245 102400          SUB    0,0      ; CREATE A ZERO
00246 041400          STA 0  0,3      ; CHANNEL NUMBER = 0
00247 020410          LDA 0  .5       ; 5 = "LOAD" FUNCTION
00250 041401          STA 0  1,3      ; SET IO FUNCTION TO "LOAD"
00251 024452          LDA 1  TEMP     ; GET NAME POINTER
00252 045403          STA 1  3,3      ; SET INTO IOCB
00253 125401          MOV    3,2      ; PASS IOCB ADDRESS
00254 006044          .IO             ; LOAD THE PROGRAM
00255 000443          JMP    BAD      ; ERROR, FAILED TO LOAD
```



```
                                
00256 030002          LDA 2  2,0      ; RESET BASE REGISTER
00257 035112          LDA 3  CB+2,2   ; LOAD BLOCK ADDRESS
00260 055010$         STA 3  RUN,2    ; NON-RESIDENT CODE
00261 007407          JSR    @T.SA,3  ; JUMP INTO IT
00262 035050  DONE:   LDA 3  RTRN,2   ; RESUME ADDRESS
00263 054437  KILL.:  STA 3  SRTN     ; SAVE RETURN
00264 021010$         LDA 0  RUN,2    ; CODE BLOCK ADDRESS
00265 101015          MOV#   0,0 SNR  ; IF ZERO,
00266 000415          JMP    OUT      ; DON'T DELETE
                                
00267 034647          LDA 3  .CB      ; INDEX TO IOCB
00270 157000          ADD    2,3      ; POINT TO IOCB
00271 041402          STA 0  2,3      ; SET INTO IOCB
00272 102400          SUB    0,0
00273 041010$         STA 0  RUN,2    ; CLEAR BLOCK ADDRESS
00274 041400          STA 0  0,3      ; CHANNEL NUMBER = 0
00275 101400          INC    0,0
00276 041401          STA 0  1,3      ; 1 = DEALLOCATE
00277 171000          MOV    3,2      ; PASS IOCB ADDRESS
00300 006044          .IO             ; DEALLOCATE BLOCK
00301 000447          ABORT           ; MUST NEVER OCCUR
00302 030002          LDA 2  2,0      ; RESTORE AC2
                                
00303 102400  OUT:    SUB    0,0
00304 041004$         STA 0  EDIT,2   ; CLEAR EDIT TABLE
00305 041011$         STA 0  END,2    ; AND END CONTROL
00306 035050          LDA 3  RTRN,2   ; RESUME ADDRESS
00307 020413          LDA 0  SRTN     ; FETCH RETURN
00310 116414          SEQ    0,3      ; IF NOT ALIKE,
00311 002411          JMP    @SRTN    ; RETURN
00312 020415          LDA 0  OFF      ; MOTOR-OFF FLAG
00313 025006$         LDA 1  HOME,2   ; CURRENT POSITION
00314 003003$         JMP    @ZAP,2   ; EXIT THRU ZAP
                                
00315 030002  BAD:    LDA 2  2,0      ; RESET BASE REGISTER
00316 020625          LDA 0  D2       ; SET OUTPUT FLAGS
00317 024407          LDA 1  ERR      ; GET ERROR MESSAGE
00320 047003$         JSR    @ZAP,2   ; INDICATE ERROR
00321 000741          JMP    DONE     ; AND CLEAR OUT
                                
00322 000000  SRTN:   0               ; SUBROUTINE RETURN ADDRESS
00323 000000  TEMP:   0               ; TEMPORARY STORAGE WORD
                                
00324 177400  MASKH:  377*400
00325 000377  MASKL:  377
00326 043477  ERR:    7*400+"?
00327 040000  OFF:    40000
00330 000137  .137:   137
00331 000132  .132:   132
00332 000100  .100:   100
00333 000071  .71:    71
00334 000060  .60:    60
00335 000040  .40:    40
00336 000014  .14:    14
00337 000005  .5:     5
                                
                      .END
```

SYMBOL	VALUE	DEF'N	REFERENCES						
	[illegible]	2:34	2:17						
	[illegible]	5:36	3:14	4:58					
LL	000061'	2:29	2:20	2:21	2:23	2:24	2:37	2:57	3:36
			3:41	3:51	3:58				
LL.	000003'	1:29	1:26						
	[illegible]	1:24	3:26	5:03					
	$X	1:10	3:38						
N	000043'	2:12	2:05						
IT	000066	1:21	1:37	1:41	2:35	2:40	2:50	2:55	3:02
			3:07	3:12	3:18	3:33	4:05	4:14	4:26
LF	000042'	2:11	2:02						
	000142'	3:30	2:30	4:02					
	000143'	3:31	2:01	5:37					
A	000144'	3:33	2:26						
	[illegible]	5:36	5:40						
	[illegible]	2:10	3:15						
	[illegible]	3:27	1:55	4:40					
	[illegible]	3:28							
	000141'	3:29	2:04						
T	$X	1:09	1:45	5:26					
	$X	1:14	1:47	5:27					
	[illegible]	5:47	5:33						
G	$X	1:12	1:49						
	[illegible]	1:23	3:25						
	[illegible]	4:12	2:46	3:19					
	$X	1:11	2:41	2:51	3:01	4:09	5:33		
.	[illegible]	5:37	1:26	4:47					
	[illegible]	5:45	4:31						
L	000325'	5:46	4:19						
T	000044'	2:13	1:50						
E	000177'	4:08	2:42	3:04	4:41				
	[illegible]	5:48	5:32						
.	000240'	4:45	1:26						
	[illegible]	5:25	5:10						
	000231'	2:01	1:52						
	[illegible]	2:04	1:57						
IT	000067	1:20	1:34	4:13	4:25				
	[illegible]	4:23	3:17	4:04					
P	000126'	3:17	3:23						
	000121'	3:11	2:22	2:48					
	[illegible]	2:54	2:19						
	[illegible]	4:02	3:55						
I	000050	1:18	1:29	2:07	2:32	2:34	2:43	2:45	2:54
			3:05	3:11	4:06	5:06	5:28		
	$X	1:13	5:04	5:08	5:16				
CE	000116'	3:07	2:25						
	[illegible]	5:42	4:12	4:21	4:23	4:38	5:07	5:29	5:31
	[illegible]	2:45	2:18	2:52					
	[illegible]	2:15	1:44						
	$X	1:06	1:54						
	[illegible]	5:43	4:24	4:33	4:46	4:54			
	000234'	4:40	3:21						
	$X	1:08	1:56	2:03	2:06	2:31	4:03	4:10	5:34
			5:39						
I	000040'	2:09	1:36						

SYMBOL	VALUE	DEF'N	REFERENCES		
.100	000332'	5:51	3:56		
.132	000331'	5:50	3:49		
.137	000330'	5:49	3:43		
.14	000336'	5:55	3:34		
.40	000335'	5:54	3:46		
.5	000337'	5:56	4:52		
.60	000334'	5:53	3:39		
.71	000333'	5:52	3:53		
.CB	000136'	3:26	4:48	5:12	
.FN	000135'	3:25	1:31	4:42	
.KILL	$X	1:05	1:30	1:46	

FLAGGED LINES: NONE

EVENT FUNCTION (MAKE.)

The event function generates a new ticket. If another ticket is current, it sets the OLD word in the terminal information table so it may be recalled. The event function asks for an incident code and a location. It fills in the following fields of the new ticket:

Time
Time zone
Date
Incident
Location
Department
Post
Operator

The event routine also creates a new entry in the status table, adds a new block to the block list, generates a disk file for the new ticket, plugs the paging function into the end control vector and plugs the format edit routine into the edit vector. It exits with a call to TAB. in the edit routine which positions the cursor in the ticket and sets things up for keyboard data input.

```
PAGE  1          A I D S     E V E N T      F U N C T I O N
02/20/74
  1                       .HEAD   A I D S     E V E N T      F U N C T I O N
  2
  3                       .TITLE  MAKE.
  4
  5                       .EXTD   .CALL   ; INVOKE ENTRY
  6
  7             000050    RC=     R0      ; RETURN ADDRESS
  8
  9                       .ENT    MAKE.
 10                       .NREL
 11
 12    00000'055050       MAKE.:  STA 3   RC,2    ; SAVE RETURN
 13    00001'006001$              JSR     @.CALL  ; CALL FUNCTION
 14    00002'000003'              .+1
 15
 16          000001       .TXTM   1
 17    00003'040511       .TXT    *AIDS.EVENT*
 18    00004'042123
 19    00005'027105
 20    00006'053105
 21    00007'047124
 22    00010'000000
 23
 24                       .END
```

SYMBOL	VALUE	DEFN	REFERENCES
[illegible]	[illegible]	1:12	1:09
[illegible]	[illegible]50	1:07	1:12
[illegible]LL	SX	1:05	1:13

FLAGGED LINES: NONE

EVENT UTILITY (EVENT)

The Event function is used to create new tickets. It displays the message "CODE" and waits for the operator to input an incident code. It then displays the message "LOCATION" and waits for the operator to input a location string. If a null location string is input, the function is terminated. The control center name is tested for validity and if it is invalid the function is terminated.

The previous ticket information table word is updated from the ticket word. If any data block is in use, its use count is reduced. The next available ticket number is put into the ticket word and decoded into a file name, with which a new disk file is created and set to all blanks.

The format write routine (PUT.) is used to fill in the time, date, incident code, operator number, post number, and department.

The post number is returned from a call to the location look-up routine (POST.). The incident field is filled in by the incident look-up routine (CODE.).

The newly created ticket is entered into the ticket status table and its data block is added to the current block list.

The console log writer is called upon to make a log entry and the format display routine is called to display the ticket on the terminal display device.


```
                .HEAD   A I D S    E V E N T    U T I L I T Y

                .TITLE  EVENT

                ;       AIDS TABLE ASSIGNMENTS

000100  .MPY=    100     ; UNSIGNED MULTIPLY
000102  .CVA=    102     ; ASCII CONVERSION
000103  .CVB=    103     ; BINARY CONVERSION
000104  TYPE=    104     ; TEST STRING TYPER
000105  ASK=     105     ; DATA STRING INPUT
000107  .PUT=    107     ; FORMATTED DATA OUTPUT
000110  .DUMP=   110     ; FORMATTED BLOCK OUTPUT
000112  .FORM=   112     ; TICKET FORMAT BLOCK
000116  .TAB=    116     ; ADVANCE TO NEXT FIELD
000123  .TSI=    123     ; TICKET STATUS INSERT
000130  .CCN=    130     ; FIND CONTROL CENTER
000137  .LOG=    137     ; SYSTEM LOG ENTRY
000140  TAD=     140     ; TAG-LINE ADDRESS
000144  LIST=    144     ; BLOCK LIST POINTER
000145  ZONE=    145     ; CURRENT TIME ZONE
000157  NEXT=    157     ; NEXT TICKET NUMBER

                ;       TIT WORD ASSIGNMENTS

000002  LINK=    2       ; GENERAL LINK WORD
000006  ZAP=     6       ; DISPLAY OUTPUT ENTRY
000010  EDIT=    10      ; EDIT TABLE ADDRESS
000015  DATA=    15      ; DATA BLOCK ADDRESS
000014  FORM=    14      ; FORMAT BLOCK ADDRESS
000016  FIELD=   16      ; FORMAT FIELD NUMBER
000017  INDEX=   17      ; FIELD POSITION INDEX
000020  CCN=     20      ; CONTROL CENTER NAME
000021  OPR=     21      ; OPERATOR NUMBER
000022  TIME=    22      ; CURRENT TIME
000023  UNIT=    23      ; CURRENT UNIT NUMBER
000024  TKT=     24      ; CURRENT TICKET NUMBER
000025  OLD=     25      ; PREVIOUS TICKET NUMBER

000040  CODE=    P0      ; INCIDENT CODE
000041  POST=    P1      ; POST NUMBER
000042  PRTY=    P2      ; PRIORITY
000043  TLIM=    P3      ; TIME LIMIT
000042  TLOG=    P2      ; TICKET NUMBER

000043  D5=      P3
000044  D4=      P4
000045  D3=      P5      ; DECIMAL DIGITS
000046  D2=      P6
000047  D1=      P7

000052  RT=      R2      ; RETURN ADDRESS

000060  DEPT=    S0      ; DEPARTMENT NAME

000100  FN=      100     ; FILENAME AREA
000114  TC=      114     ; INCIDENT AREA
000124  TL=      124     ; LOCATION AREA
```

```
                        .NREL

                        ; REQUEST PARAMETERS

                EVENT:  STA     3,RT,2  ; SAVE RETURN
                        SUB     0,0
                        LDA 1   TAD     ; POSITION
                        JSR     @ZAP,2

                        JSR     @TYPE   ; TYPE "CODE"
                        MSG1
                        LDA     0,.4
                        LDA     1,.TC   ; INCIDENT OFFSET
                        ADD     2,1     ; MAKE ADDRESS
                        JSR     @ASK    ; GET INCIDENT
                        NOP             ; IF NONE, OKAY

                        JSR     @TYPE   ; TYPE "LOCATION"
                        MSG2
                        LDA     0,.30
                        LDA     1,.TL   ; LOCATION OFFSET
                        ADD     2,1     ; MAKE ADDRESS
                        JSR     @ASK    ; GET LOCATION
                        JMP     @RT,2   ; IF NONE, EXIT

                        ; TEST FOR VALIDITY

                        LDA 0   CCN,2   ; CONTROL CENTER
                        JSR     @.CCN   ; LOCATE IN TABLE
                        JMP     @RT,2   ; NOT IN TABLE
                        LDA 1   1,3     ; GET DEPT NAME
                        STA 1   DEPT,2  ; AND SAVE IT

                        ; CLEAR OUT OLD TICKET

                        LDA 3   TKT,2   ; CURRENT TICKET
                        STA 3   OLD,2   ; BECOMES OLD ONE

                        LDA 3   DATA,2  ; DATA BLOCK
                        MOV#    3,3 SZR ; IF ANY DATA,
                        DSZ     USE,3   ; REDUCE USE-COUNT
                        JMP     ALL     ; CONTINUE

                        STA 3   LIST    ; NEXT IN LINE
                        IDLE            ; IN CASE OF SKIP
```



```
                        ; ALLOCATE BLOCK SPACE

000037'102520   ALL:    SUBZL   0,0     ; TYPE = 1
000040'040512           STA 0   BLOCK   ; SET TYPE
000041'030506           LDA 2   .IOCB   ; IOCB ADDRESS
000042'006044           .IO             ; ALLOCATE
000043'000773           JMP     ALL-1   ; TRY AGAIN

000044'030002           LDA 2   2       ; RESTORE AC2
000045'024505           LDA 1   BLOCK   ; BLOCK ADDRESS

                        ; WAIT FOR THE DISK

000046'102520           SUBZL   0,0     ; DISK = 1
000047'004050           ENQ             ; GRAB IT

000050'045015           STA 1   DATA,2  ; PUT INTO TIT
000051'020510           LDA 0   .FWB    ; BUFFER OFFSET
000052'107000           ADD     0,1     ; BUFFER ADDRESS
000053'044514           STA 1   SAVE    ; SAVE FOR LATER

                        ; CONSTRUCT FILE NAME

000054'010157           ISZ     NEXT    ; BUMP TICKET NUMBER
000055'024157           LDA 1   NEXT    ; NEXT TICKET NUMBER
000056'045024           STA 1   TKT,2   ; PUT IT IN TIT
000057'006102           JSR     @.CVA   ; MAKE ASCII

000060'021043           LDA 0   D5,2    ; DIGIT FIVE
000061'101300           MOVS    0,0     ; TO TOP BYTE
000062'025044           LDA 1   D4,2    ; DIGIT FOUR
000063'123000           ADD     1,0     ; COMBINE
000064'041100           STA 0   F1,2    ; AND SAVE

000065'021045           LDA 0   D3,2    ; DIGIT THREE
000066'101300           MOVS    0,0     ; TO TOP BYTE
000067'025046           LDA 1   D2,2    ; DIGIT TWO
000070'123000           ADD     1,0     ; COMBINE
000071'041101           STA 0   F2,2    ; AND SAVE

000072'021047           LDA 0   D1,2    ; DIGIT ONE
000073'101300           MOVS    0,0     ; TO TOP BYTE
000074'041102           STA 0   F3,2    ; AND SAVE
```

```
                        ; CREATE A NEW FILE

00075'024475                    LDA 1   .FN     ; DATA OFFSET
00076'133003                    ADD     1,2     ; FILE NAME
00077'034041                    LDA 3   DDOS    ; SYSTEM TABLE
00100'024473                    LDA 1   TYPES   ; FILE TYPE BITS
00101'020473                    LDA 0   PROPS   ; AND PROPERTIES
00102'037403                    JSR     @D.CYS,3; CALL OVERLAY
00103'000014                    0.CRT           ; CREATE A FILE

                        ; READ IN A SEGMENT

00104'021010                    LDA 0   F.ID,2  ; FILE IDENTIFIER
00105'041454                    STA 0   D.SP3,3 ; SAVE FOR PRIME
00106'021013                    LDA 0   F.FST,2 ; HIGH-ORDER ADDRESS
00107'025015                    LDA 1   F.BFP,2 ; LOW-ORDER ADDRESS
00110'034455                    LDA 2   .377    ; HALF-WORD MASK
00111'101220                    MOVZR   0,0     ; SHIFT HIGH-ORDER, THEN
00112'147530                    ANDL    2,1     ; LOW-ORDER, WITH CARRY

00113'030454                    LDA 2   SAVE    ; BUFFER ADDRESS
00114'035753                    JSR     D.PRM,3 ; PRIME THE BUFFER
00115'030002                    LDA 2   2       ; RESTORE AC2

                        ; INDICATE SECTOR FULL

00116'034451                    LDA 3   SAVE    ; BUFFER ADDRESS
00117'021407                    LDA 0   B.FOR,3 ; FORE POINTER
00120'110134                    COMZL#  0,0 SZR ; IF NOT NULL,
00121'000407                    JMP     OK      ; LEAVE IT ALONE

00122'021411                    LDA 0   B.BFP,3 ; BACK/FORE POINTERWORD
00123'024443                    LDA 1   .177400 ; HIGH-BYTE MASK
00124'123003                    AND     1,0     ; SAVE BACK HALF
00125'024421                    LDA 1   FULL    ; = 250.
00126'123000                    ADD     1,0     ; ADD BYTE LIMIT
00127'041411                    STA 0   B.BFP,3 ; REPLACE POINTERWORD

                        ; CLEAR TO ALL-BLANKS

00130'020432    OK:     LDA 0   .FWD    ; DATA OFFSET
00131'117000            ADD     0,3     ; MOVE TO START
00132'020412            LDA 0   COUNT   ; BLANK COUNTER
00133'024410            LDA 1   SPACE   ; ASCII SPACE-CODE
00134'045400            STA 1   0,3     ; CLEAR A WORD
00135'175400            INC     3,3     ; BUMP ADDRESS
00136'101404            INC     0,0 SZR ; AND COUNTER
00137'000775            JMP     .-3     ; IF MORE, REPEAT

                        ; RELEASE THE DISK

00140'102520            SUBZL   0,0     ; DISK = 1
00141'004052            DEQ             ; GIVE IT UP
00142'000433            JMP     FILL    ; CONTINUE
```



```
000100          F1=     FN
000101          F2=     FN+1    ; FILE NAME
000102          F3=     FN+2

000060          HOURS=  60      ; DINOS HOURS WORD
000061          MINS=   61      ; DINOS MINUTES WORD
000064          DAY=    64
000065          MONTH=  65      ; DINOS DATE WORDS
000066          YEAR=   66

00143'020040    SPACE:  " *401
00144'177411    COUNT:  B.FWD+3-B.LEN
00145'000467'   .CAL:   CAL-2
00146'000372    FULL:   500./2

00147'000150'   .IOCB:  .+1     ; ALLOCATE IOCB
00150'000000            0
00151'000000            0
00152'000000    BLOCK:  0
00153'000000            0
00154'000420            T.FW+B.LEN+1

00155'000004    .4:     4
00156'000017    .17:    17
00157'177750    .30:    -30
00160'000124    ..T:    "T
00161'000010    .FWB:   T.FW
00162'000020    .FWD:   B.FWD+3
00163'000144    .100.:  100.
00164'001750    .TEN.:  1000.
00165'000377    .377:   377     ; LO-BYTE MASK
00166'177400    .1774:  377*400 ; HI-BYTE MASK
00167'000000    SAVE:   0       ; ADDRESS SAVE

00170'000114    .TC:    TC      ; INCIDENT OFFSET
00171'000124    .TL:    TL      ; LOCATION OFFSET
00172'000100    .FN:    F1      ; FILE-NAME POINTER
00173'000001    TYPES:  TYP.B   ; FILE TYPE BINARY
00174'000400    PROPS:  PRP.N   ; NAME-PROTECTED

                ; DATA BLOCK ASSIGNMENTS

        000007  MOD=    7       ; DATA BLOCK UPDATE FLAG
        000006  USE=    6       ; DATA BLOCK USAGE COUNT
        000005  BLK=    5       ; BLOCK NUMBER IN TICKET
        000004  REF=    4       ; TICKET NUMBER IN BLOCK

                ; FORMAT BLOCK FIELD INDICES

        177776  FIC=    -2      ; INCIDENT CODE
        177774  FIP=    -4      ; INCIDENT POST
        177773  FIT=    -5      ; INCIDENT TIME
        177767  FRB=    -11     ; RECEIVED BY
        177761  FPD=    -17     ; DEPARTMENT
```

```
                    ; SET UP TO INSERT DATA

00175'034112  FILL:   LDA 3   .FORM   ; TICKET FORMAT BLOCK
00176'055014          STA 3   FORM,2  ; PUT IT IN THE TIT
00177'035017          LDA 3   T.SA,3  ; START OF FORMAT
00200'025773          LDA 1   FIT,3   ; FIELD INDEX
00201'045016          STA 1   FIELD,2 ; INCIDENT TIME
00202'132400          SUB     0,0
00203'041017          STA 0   INDEX,2 ; POSITION ZERO

                    ; CONVERT THE TIME

00204'024063          LDA 1   HOURS   ; GET THE HOURS
00205'020756          LDA 0   .100.
00206'006133          JSR     @.MPY   ; TIMES 100.
00207'020061          LDA 0   MINS    ; GET THE MINUTES
00210'107000          ADD     0,1     ; ADD TOGETHER
00211'045022          STA 1   TIME,2  ; SAVE FOR LATER
00212'006102          JSR     @.CVA   ; MAKE ASCII

00213'025044          LDA 1   D4,2    ; DIGIT FOUR
00214'006107          JSR     @.PUT
00215'025045          LDA 1   D3,2    ; DIGIT THREE
00216'006107          JSR     @.PUT
00217'025046          LDA 1   D2,2    ; DIGIT TWO
00220'006107          JSR     @.PUT
00221'025047          LDA 1   D1,2    ; DIGIT ONE
00222'006107          JSR     @.PUT

00223'024145          LDA 1   ZONE    ; TIME ZONE
00224'125300          MOVS    1,1     ; 1ST CHARACTER
00225'006107          JSR     @.PUT
00226'024145          LDA 1   ZONE    ; 2ND CHARACTER
00227'006107          JSR     @.PUT
00230'024133          LDA 1   ..T     ; LITERAL "T"
00231'006107          JSR     @.PUT
```



```
1                   ; CONVERT THE DATE
2
3     00232'024064          LDA 1   DAY     ; GET THE DAY
4     00233'006102          JSR     @.CVA   ; MAKE ASCII
5     00234'025046          LDA 1   D2,2    ; DIGIT TWO
6     00235'020721          LDA 0   .17     ; 4-BIT MASK
7     00236'107415          AND#    0,1 SNR ; IF ASCII ZERO,
8     00237'106000          ADC     0,1     ; FORCE A SPACE
9     00240'006107          JSR     @.PUT
10    00241'025047          LDA 1   D1,2    ; DIGIT ONE
11    00242'006107          JSR     @.PUT
12
13    00243'024065          LDA 1   MONTH   ; GET THE MONTH
14    00244'125120          MOVZL   1,1     ; MULTIPLY BY TWO
15    00245'034741          LDA 3   .CAL    ; CALENDAR BLOCK
16    00246'137000          ADD     1,3     ; INDEX INTO IT
17    00247'025401          LDA 1   1,3     ; NAME OF MONTH
18    00250'125300          MOVS    1,1     ; 3RD CHARACTER
19    00251'045047          STA 1   D1,2
20    00252'025400          LDA 1   0,3     ; 2ND CHARACTER
21    00253'045046          STA 1   D2,2
22    00254'125300          MOVS    1,1     ; 1ST CHARACTER
23    00255'006107          JSR     @.PUT
24    00256'025046          LDA 1   D2,2    ; 2ND CHARACTER
25    00257'006107          JSR     @.PUT
26    00260'025047          LDA 1   D1,2    ; 3RD CHARACTER
27    00261'006107          JSR     @.PUT
28
29    00262'024066          LDA 1   YEAR    ; GET THE YEAR
30    00263'006102          JSR     @.CVA   ; CONVERT IT
31    00264'025046          LDA 1   D2,2    ; DIGIT TWO
32    00265'006107          JSR     @.PUT
33    00266'025047          LDA 1   D1,2    ; DIGIT ONE
34    00267'006107          JSR     @.PUT
```

```
                ; COMPUTE INCIDENT
                ;
00327'035014            LDA 3   FORM,2  ; TICKET FORMAT
00330'035007            LDA 3   T.SA,3  ; START OF FORMAT
00331'025776            LDA 1   FIC,3   ; CODE FIELD
00332'045016            STA 1   FIELD,2 ; INCIDENT CODE
00333'126400            SUB     1,1     ; POSITION ZERO
00334'045017            STA 1   INDEX,2
00335'045043            STA 1   D5,2    ; NO DIGIT 5
                ;
00336'025114            LDA 1   TC,2
00337'045045            STA 1   D3,2    ; DIGIT 3
00340'125300            MOVS    1,1
00341'045044            STA 1   D4,2    ; AND 4
                ;
00342'025115            LDA 1   TC+1,2
00343'045047            STA 1   D1,2    ; DIGIT 1
00344'125300            MOVS    1,1
00345'045046            STA 1   D2,2    ; AND 2
                ;
00346'006103            JSR     @.CVB   ; GO BINARY
00347'024063            LDA Ø   .100.   ; GET THRESHOLD
00350'135004            MOV     1,3 SZR ; IF NOT ZERO,
00351'031652            LDA 3   .TEN.   ; GET ADJUSTMENT
00352'046512            SLE     Ø,1     ; IF < 100.
00353'[illegible]       ADD     3,1     ; ADD 1000.
00354'[illegible]       STA 1   CODE,2  ; SAVE CODE
00355'006102            JSR     @.CVA   ; GO ASCII
                ;
00356'025044            LDA 1   D4,2    ; DIGIT 4
00357'[illegible]       JSR     SEND
00360'[illegible]       LDA 1   D3,2    ; DIGIT 3
00361'004543            JSR     SEND
00362'025046            LDA 1   D2,2    ; DIGIT 2
00363'004536            JSR     SEND
00364'025047            LDA 1   D1,2    ; DIGIT 1
00365'004534            JSR     SEND
```



```
1                       ; LOOK UP LOCATION
2
3       00327'016541            JSR     @.POST
4
5                       ; LOOK UP INCIDENT
6
7       00330'006537            JSR     @.CODE
8
9                       ; ADD TO STATUS TABLE
10
11      00331'021020            LDA Ø   CCN,2   ; CONTROL CENTER
12      00332'025024            LDA 1   TKT,2   ; TICKET NUMBER
13      00333'006123            JSR     @.TSI   ; MAKE AN ENTRY
14      00334'000424            JMP     NOT     ; NO ROOM IN TABLE
15
16      00335'025040            LDA 1   CODE,2  ; INCIDENT CODE
17      00336'045402            STA 1   2,3     ; INTO THE TABLE
18      00337'025022            LDA 1   TIME,2  ; GET THE TIME
19      00340'045403            STA 1   3,3     ; INTO THE TABLE
20      00341'025041            LDA 1   POST,2  ; GET THE POST
21      00342'045404            STA 1   4,3     ; INTO THE TABLE
22      00343'025042            LDA 1   PRTY,2  ; GET PRIORITY
23      00344'045406            STA 1   6,3     ; INTO THE TABLE
24      00345'025043            LDA 1   TLIM,2  ; GET TIME LIMIT
25      00346'045407            STA 1   7,3     ; INTO THE TABLE
26
27                      ; INSERT LOCATION
28
29      00347'024610            LDA 1   .30     ; STRING LENGTH
30      00350'125240            MOVOR   1,1     ; MAKE WORD COUNT
31      00351'021124            LDA Ø   TL,2    ; OBTAIN A WORD
32      00352'041424            STA Ø   24,3    ; AND INSERT IT
33      00353'175401            INC     3,3     ; BUMP ADDRESS
34      00354'151400            INC     2,2     ; THIS ONE TOO
35      00355'125404            INC     1,1 SZR ; BUMP COUNTER
36      00356'000773            JMP     .-5     ; AND REPEAT
37      00357'030002            LDA 2   2       ; RESTORE TIT
```

```
                ; INSERT OPERATOR

          NOT:  LDA 1   OPR,2   ; OPERATOR NUMBER
                JSR     @.CVA   ; MAKE ASCII
                LDA 3   FORM,2  ; TICKET FORMAT
                LDA 3   T.SA,3  ; START OF FORMAT
                LDA 1   FRB,3   ; RECEIVED FIELD
                STA 1   FIELD,2 ; INSTALL IN TIT
                SUB     0,0
                STA 0   INDEX,2 ; POSITION ZERO

                LDA 1   D4,2    ; DIGIT 4
                JSR     SEND
                LDA 1   D3,2    ; DIGIT 3
                JSR     SEND
                LDA 1   D2,2    ; DIGIT 2
                JSR     SEND
                LDA 1   D1,2    ; DIGIT 1
                JSR     SEND

                ; INSERT POST NUMBER

                LDA 1   POST,2  ; POST NUMBER
                JSR     @.CVA   ; MAKE ASCII
                LDA 3   FORM,2  ; TICKET FORMAT
                LDA 3   T.SA,3  ; START OF FORMAT
                LDA 1   FIP,3   ; POST FIELD
                STA 1   FIELD,2 ; INSTALL IN TIT
                SUB     0,0
                STA 0   INDEX,2 ; POSITION ZERO

                LDA 1   D4,2    ; DIGIT 4
                JSR     SEND
                LDA 1   D3,2    ; DIGIT 3
                JSR     SEND
                LDA 1   D2,2    ; DIGIT 2
                JSR     SEND
                LDA 1   D1,2    ; DIGIT 1
                JSR     SEND

                ; INSERT DEPARTMENT

                LDA 3   FORM,2  ; TICKET FORMAT
                LDA 3   T.SA,3  ; START OF FORMAT
                LDA 1   FPD,3   ; DEPARTMENT FIELD
                STA 1   FIELD,2 ; INSTALL IN TIT
                SUB     0,0
                STA 0   INDEX,2 ; POSITION ZERO

                LDA 1   DEPT,2  ; DEPARTMENT NAME
                MOVS    1,1     ; TOP BYTE FIRST
                JSR     @.PUT   ; INTO THE TICKET
                LDA 1   DEPT,2  ; THEN THE OTHER
                JSR     @.PUT   ; JUST THE SAME
```



```
                ; ADD TO BLOCK LIST

                LDA 3   DATA,2  ; BLOCK ADDRESS
                LDA 2   LIST    ; LIST POINTER
                LDA 1   LINK,2  ; FOLLOW LINK
                MOV#    2,0 SNR ; IF NO LIST,
                MOV     3,2 SKP ; MAKE ONE

                STA 1   LINK,3  ; LINK TO REAR
                STA 3   LINK,2  ; AND TO FRONT
                STA 3   LIST    ; UPDATE LIST
                LDA 2   2       ; RESTORE AC2

                LDA 0   TKT,2   ; TICKET NUMBER
                STA 0   REF,3   ; FOR REFERENCE
                STA 0   TLOG,2  ; SAVE FOR LOG
                SUB     0,0
                STA 0   BLK,3   ; BLOCK IS ZERO
                STA 0   UNIT,2  ; CLEAR FOR LOG
                SUBZL   0,0     ; USERS = 1
                STA 0   USE,3   ; SET USAGE-COUNT
                STA 0   MOD,3   ; AND UPDATE FLAG

                ; FINISH UP AND EXIT

                JSR     @.LOG   ; MAKE LOG ENTRY
                JSR     @.DUMP  ; CALL DISPLAY
                ISZ     RT,2    ; PREPARE TO SKIP
                LDA 3   .TAB    ; RESUME ADDRESS
                JMP     @RT,2   ; SKIP ON RETURN
```

```
                        ; SUPRESS LEADING ZEROES

                SEND:   LDA Ø   D5,2
                        SNE     Ø,1
                        JMP     Ø,3
                        DSZ     D5,2
                        JMP     @.PUT

                .EXTN   CODE.,POST.

                .CODE:  CODE.   ; INCIDENT LOOK-UP
                .POST:  POST.   ; LOCATION LOOK-UP

                .TXTN   1
                .TXTX   1

                CAL:    ; CALENDAR

                        .TXT    *JAN*
                        .TXT    *FEB*
                        .TXT    *MAR*
                        .TXT    *APR*
                        .TXT    *MAY*
                        .TXT    *JUN*
                        .TXT    *JUL*
                        .TXT    *AUG*
                        .TXT    *SEP*
                        .TXT    *OCT*
                        .TXT    *NOV*
                        .TXT    *DEC*

                MSG1:   .TXT    *CODE*
                MSG2:   .TXT  *  LOCATION*

                        .END    EVENT
```


SYMBOL	VALUE	DEF'N	REFERENCES						
ALL	000037'	3:03	2:42	3:07					
ASK	000105	1:11	2:15	2:23					
BLK	000005	5:45	11:18						
BLOCK	000152'	5:19	3:04	3:10					
CAL	000471'	12:17	5:13						
CCN	000020	1:33	2:28	9:11					
CODE	000040	1:40	8:27	9:16					
CODE.	'X	12:09	12:11						
COUNT	000144'	5:12	4:43						
D1	000047	1:50	3:41	6:27	7:10	7:19	7:26	7:33	8:17
			8:36	10:18	10:38				
D2	000046	1:49	3:37	6:25	7:05	7:21	7:24	7:31	8:19
			8:34	10:16	10:36				
D3	000045	1:48	3:35	6:23	8:12	8:32	10:14	10:34	
D4	000044	1:47	3:31	6:21	8:14	8:30	10:12	10:32	
D5	000043	1:46	3:29	8:09	12:03	12:06			
DATA	000015	1:29	2:39	3:17	11:03				
DAY	000064	5:07	7:03						
DEPT	000060	1:54	2:32	10:50	10:53				
EDIT	000013	1:28							
EVENT	000000'	2:05	12:35						
F1	000100	5:01	3:33	5:37					
F2	000101	5:02	3:39						
F3	000102	5:03	3:43						
FIC-	177776	5:50	8:05						
FI_	000016	1:31	6:07	8:06	10:08	10:28	10:46		
FILE	000175'	6:03	4:54						
FIP	177774	5:51	10:27						
FIT	177773	5:52	6:06						
FN	000100	1:56	5:31	5:02	5:03				
FORM	000014	1:30	6:04	8:03	10:05	10:25	10:43		
FPD	177761	5:54	10:45						
FRB	177767	5:53	10:07						
FULL	000146'	5:14	4:35						
HOURS	000064	5:05	6:13						
INDEX	000017	1:32	6:09	8:08	10:10	10:30	10:48		
LINK	000002	1:26	11:05	11:09	11:10				
LIST	000144	1:20	2:44	11:04	11:11				
MINS	000061	5:06	6:16						
MOD	000007	5:43	11:22						
MONTH	000065	5:08	7:13						
MSG1	000521'	12:32	2:11						
MSG2	000524'	12:33	2:19						
NEXT	000157	1:22	3:24	3:25					
NOT	000360'	10:03	9:14						
NX	000130'	4:41	4:30						
OLD	000025	1:38	2:37						
OPR	000021	1:34	10:03						
POST	000041	1:41	9:20	10:23					
POST.	'X	12:39	12:12						
PRO	000174'	5:39	4:07						
PR	000042	1:42	9:22						
PUF	000004	5:46	11:15						
RT	000052	1:52	2:05	2:24	2:30	11:28	11:30		
SAVE	000167'	5:33	3:20	4:21	4:27				

A I D S E V E N T U T I L I T Y

L	VALUE	DEF'N	REFERENCES						
		12:03	8:31	8:33	8:35	8:37	10:13	10:15	10:17
			10:19	10:33	10:35	10:37	10:39		
		5:11	4:44						
		1:19	2:07						
		1:57	5:35	8:11	8:16				
		1:35	6:18	9:18					
		1:37	2:36	3:26	9:12	11:14			
		1:24	5:36	9:31					
		1:43	9:24						
		1:44	11:16						
		1:19	2:10	2:18					
		5:38	4:46						
		1:36	11:19						
		5:44	2:41	11:21					
		5:39	7:22						
		1:27	2:38						
		1:21	6:33	6:33					
		5:29	6:14	8:22					
		5:24	7:15						
		5:32	4:33						
		5:25	2:29	9:29					
		5:31	4:17						
		5:23	2:12						
		5:13	7:15						
		1:17	2:29						
		12:11	9:07						
		1:12	3:27	6:19	7:04	7:30	8:28	10:04	10:24
		1:19	8:21						
		1:13	11:27						
		5:37	4:03						
		1:14	6:03						
		5:27	3:18						
		5:28	4:41						
		5:16	3:05						
		1:18	11:26						
		1:37	6:15						
		12:12	9:03						
		1:12	6:22	6:24	6:26	6:28	6:32	6:34	6:36
			7:09	7:11	7:23	7:25	7:27	7:32	7:34
			10:52	10:54	12:07				
		1:15	11:29						
		5:35	2:13						
		5:30	8:24						
		5:36	2:21						
		1:16	9:13						
		5:26	6:35						

FLAGGED LINES: NONE

INCIDENT LOOKUP (CODE.)

Incident lookup is called by the event function. It is passed an incident code (ten code) through a parameter word in the terminal information table, and reads the file "INCIDENT.CODES" looking for a matching entry. If one is found, it copies a text string description from the file into the ticket data block and returns the time limit and priority assigned to the code through two more parameter words. If no matching file entry is found, the priority and time limit are set to zero and the text field is left blank.


```
                        .HEAD   A I D S    I N C I D E N T    L O O K U P
                        .TITLE  CODE.

          000107        .PUT=   107     ; FORMATTED OUTPUT

          000014        FORM=   14      ; FORMAT BLOCK ADDRESS
          000016        FIELD=  16      ; FORMAT FIELD NUMBER
          000017        INDEX=  17      ; FIELD POSITION INDEX

          000040        CODE=   P0      ; WANTED INCIDENT CODE
          000042        PRTY=   P2      ; PRIORITY OF INCIDENT
          000043        TLIM=   P3      ; WATCH-DOG TIMER LIMIT

          000062        BUF=    S2      ; BUFFER ADDRESS
          000063        THIS=   S3      ; CURRENT INCIDENT CODE

          000051        RTRN=   R1      ; RETURN ADDRESS

          177777        FIN=    -1      ; INCIDENT FIELD INDEX

                        .ENT    CODE.
                        .NREL

000000'055051   CODE.:  STA 3   RTRN,2  ; SAVE RETURN ADDRESS
000001'102400           SUB     0,0     ; CREATE A ZERO
000002'041042           STA 0   PRTY,2  ; CLEAR PRIORITY
000003'041043           STA 0   TLIM,2  ; AND TIME LIMIT

000004'030530           LDA 2   .IOC1   ; GET IOCB ADDRESS
000005'000044           .IO             ; ALLOCATE SPACE FOR BUFFER
000006'000475           JMP     FAIL    ; ERROR RETURN IF NO CAN DO
000007'030002           LDA 2   2,0     ; RESTORE AC2

000010'024527           LDA 1   BLK1    ; GET BLOCK ADDRESS
000011'102520           SUBZL   0,0     ; DISK = 1
000012'004050           ENQ             ; WAIT FOR IT
000013'044532           STA 1   BLK2    ; SAVE BLOCK ADDRESS
000014'020477           LDA 0   .FWD    ; BUFFER DISPLACEMENT
000015'107000           ADD     0,1     ; DISPLACE INTO BUFFER
000016'045062           STA 1   BUF,2   ; SAVE BUFFER ADDRESS

000017'034041           LDA 3   DDOS    ; GET SYSTEM ADDRESS
000020'030503           LDA 2   FILE    ; POINT TO FILE NAME
000021'007402           JSR     @D.SRC,3; SEARCH FOR FILE
000022'000461           JMP     FAIL    ; FILE NOT FOUND

000023'021010           LDA 0   F.ID,2  ; GET FILE IDENTIFIER
000024'041454           STA 0   D.SP3,3 ; SET UP PARAMS FOR PRIME
000025'021013           LDA 0   F.FST,2 ; HIGH ORDER DISK ADDRESS
000026'025015           LDA 1   F.BEP,2 ; LOW ORDER DISK ADDRESS
000027'030473           LDA 2   .377    ; HALF WORD MASK
000030'101220           MOVZR   0,0     ; SHIFT HIGH ORDER ADR, THEN
000031'117500           ANDL    2,1     ; LOW ORDER, WITH CARRY
000032'030002           LDA 2   2,0     ; SET UP BASE REGISTER
000033'031062           LDA 2   BUF,2   ; BUFFER PREFIX ADDRESS
000034'005753           JSR     D.PRM,3 ; PRIME INPUT BUFFER
```

```
                LDA 2   2,0     ; RESTORE BASE REGISTER
                LDA 3   FORM,2  ; FORMAT BLOCK
                LDA 3   T.SA,3  ; START OF FORMAT
                LDA 1   FIN,3   ; INCIDENT FIELD
                STA 1   FIELD,2 ; SET FIELD NUMBER
                SUB     0,0     ; CREATE A ZERO
                STA 0   INDEX,2 ; CHARACTER INDEX

INIT:           SUB     0,0     ; CREATE A ZERO
                STA 0   THIS,2  ; CLEAR CODE WORD

RDLP:           LDA 3   DDOS    ; GET SYSTEM ADDRESS
                LDA 2   BUF,2   ; GET BUFFER ADDRESS
                JSR     @D.IN,3 ; READ ONE CHARACTER
                JMP     DONE    ; DONE IF END OF FILE
                LDA 1   .177    ; FETCH MASK
                AND     0,1     ; EXTRACT CHARACTER
                LDA 2   2,0     ; RESTORE AC2

                LDA 0   SEMI    ; SEMI-COLON
                SNE     0,1     ; COMPARE
                JMP     XLP     ; SKIP COMMENT

                LDA 0   COLON   ; ASCII COLON
                SNE     0,1     ; COMPARE
                JMP     TEST    ; CODE IS COMPUTED

                LDA 3   THIS,2  ; GET PARTIAL CODE
                MOVZL   3,0     ; TIMES TWO
                MOVZL   0,0     ; TIMES FOUR
                ADDZL   0,3     ; TIMES TEN

                LDA 0   .60     ; ASCII ZERO
                SUB     0,1     ; CONVERT TO BINARY
                ADD     1,3     ; ADD TO CODE
                STA 3   THIS,2  ; STORE NEW CODE
                JMP     RDLP    ; GET NEXT CHARACTER

TEST:           LDA 0   CODE,2  ; GET WANTED CODE
                LDA 1   THIS,2  ; AND THIS CODE
                SNE     0,1     ; COMPARE
                JMP     OUT     ; OUTPUT INCIDENT

XLP:            LDA 3   DDOS    ; GET SYSTEM ADDRESS
                LDA 2   BUF,2   ; AND BUFFER ADDRESS
                JSR     @D.IN,3 ; READ ONE CHARACTER
FAIL:           JMP     DONE    ; ERROR EXIT
                LDA 1   .177    ; FETCH MASK
                AND     0,1     ; EXTRACT CHARACTER
                LDA 2   2,0     ; RESTORE AC2

                LDA 0   .LF     ; ASCII RETURN
                SEQ     0,1     ; END OF ENTRY?
                JMP     XLP     ; NO, SKIP SOME MORE
                JMP     INIT    ; START ON NEXT RECORD

.FWD:           T.FW    ; BUFFER DISPLACEMENT
.LF:            12      ; ASCII LINE-FEED
```



```
00115 000015 .CR:    15      ; CARRIAGE RETURN
00116 000060 .60:    60      ; CHARACTER ZERO
00117 000072 COLON:  ":      ; ASCII COLON
00120 000073 SEMI:   ";      ; SEMI-COLON
00121 000177 .177:   177     ; 7-BIT MASK
00122 000377 .377:   377     ; 8-BIT MASK
00123 000124 FILE:   .+1     ; FILE NAME

             .TXTM   1
             .TXTX   1
00124 044516 .TXT    *INCIDENT.CODES*

00134 000135 .IOC1:  .+1     ; IOC TO ALLOCATE

00135 000004         0       ; CHANNEL
00136 000000         0       ; FUNCTION
00137 000000 BLK1:   0       ; ADDRESS
00140 000000         0
00141 000420         T.FW+B.LEN+1

00142 000143 .IOC2:  .+1     ; IOC TO DEALLOCATE

00143 000003         0       ; CHANNEL
00144 000001         1       ; FUNCTION
00145 000000 BLK2:   0       ; ADDRESS

00146 034041 OUT:    LDA 3   DDOS    ; GET SYSTEM ADDRESS
00147 031062         LDA 2   BUF,2   ; GET BUFFER ADDRESS
00150 017404         JSR     @D.IN,3 ; READ ONE CHARACTER
00151 000457         JMP     DONE    ; ERROR EXIT
00152 024747         LDA 1   .177    ; FETCH MASK
00153 104004         AND     0,1     ; EXTRACT CHARACTER INTO AC1
00154 031002         LDA 2   2,0     ; RESTORE AC2

00155 020742         LDA 0   COLON   ; ASCII COLON
00156 106415         SNE     0,1     ; COMPARE
00157 000403         JMP     IDON    ; INCIDENT IS OUT

00160 006107         JSR     @.PUT   ; OUTPUT CHARACTER
00161 000765         JMP     OUT     ; GET NEXT CHARACTER

00162 034041 IDON:   LDA 3   DDOS    ; GET SYSTEM ADDRESS
00163 031062         LDA 2   BUF,2   ; AND BUFFER ADDRESS
00164 017404         JSR     @D.IN,3 ; READ ONE CHARACTER
00165 000443         JMP     DONE    ; ERROR EXIT
00166 024733         LDA 1   .177    ; FETCH MASK
00167 104004         AND     0,1     ; EXTRACT CHARACTER
00170 031002         LDA 2   2,0     ; RESTORE AC2

00171 020726         LDA 0   COLON   ; ASCII COLON
00172 106415         SNE     0,1     ; COMPARE
00173 000412         JMP     PDON    ; PRIORITY IS OUT

00174 035042         LDA 3   PRTY,2  ; GET PRIORITY
00175 161120         MOVZL   3,0     ; TIMES TWO
00176 141124         MOVZL   0,4     ; TIMES FOUR
00177 117120         ADDZL   0,3     ; TIMES TEN
```

```
                        LDA 0   .60     ; ASCII ZERO
                        SUB     0,1     ; CONVERT TO BINARY
                        ADD     1,3     ; ADD TO PRIORITY
                        STA 3   PRTY,2  ; STORE NEW PRIORITY
                        JMP     IDON    ; READ NEXT CHARACTER

                PDON:   LDA 3   DDOS    ; GET SYSTEM ADDRESS
                        LDA 2   BUF,2   ; AND BUFFER ADDRESS
                        JSR     @D.IN,3 ; READ ONE CHARACTER
                        JMP     DONE    ; ERROR EXIT
                        LDA 1   .177    ; FETCH MASK
                        AND     0,1     ; EXTRACT CHARACTER
                        LDA 2   2,0     ; RESTORE AC2

                        LDA 0   .CR     ; ASCII RETURN
                        SNE     0,1     ; COMPARE
                        JMP     DONE    ; ALL DONE

                        LDA 3   TLIM,2  ; TIME LIMIT
                        MOVZL   3,3     ; TIMES TWO
                        MOVZL   0,0     ; TIMES FOUR
                        ADDZL   0,3     ; TIMES TEN

                        LDA 0   .60     ; ASCII ZERO
                        SUB     0,1     ; CONVERT TO BINARY
                        ADD     1,3     ; ADD TIME LIMIT
                        STA 3   TLIM,2  ; STORE NEW LIMIT
                        JMP     PDON    ; READ NEW CHARACTER

                DONE:   LDA 2   .IOC2   ; AND IOCB ADDRESS
                        .IO             ; DEALLOCATE BUFFER
                        ABORT           ; FATAL IF NO CAN DO
                        LDA 2   2,0     ; RESTORE AC2
                        SUBZL   0,0     ; DISK = 1
                        DEC             ; GIVE IT UP
                        JMP     @RTRN,2 ; RETURN

                .END
```


SYMBOL	VALUE	DEF'N	REFERENCES					
BLK1	000137'	3:17	1:35					
BLK2	000145'	3:25	1:38					
BUF	000062	1:15	1:41	1:56	2:13	2:45	3:28	3:43	4:[illegible]
CODE	000040	1:11	2:39					
CODE.	000004'	1:25	1:22					
COLON	000117'	3:03	2:24	3:35	3:50			
DONE	000230'	4:30	2:15	2:47	3:30	3:45	4:10	4:17	
FAIL	000103'	2:47	1:32	1:46				
FIELD	000016	1:08	2:05					
FILE	000123'	3:07	1:44					
FIN	177777	1:20	2:34					
FORM	000014	1:07	2:02					
IDON	000162'	3:42	3:37	4:05				
INDEX	000017	1:09	2:07					
INIT	000044'	2:09	2:55					
OUT	000146'	3:27	2:42	3:40				
PDON	000205'	4:17	3:52	4:28				
PRTY	000042	1:12	1:27	3:54	4:04			
PULP	000046'	2:12	2:37					
PULS	000051	1:18	1:25	4:36				
QUIT	000120'	3:04	2:23					
QUOT	000074'	2:39	2:26					
TIME	000063	1:16	2:11	2:28	2:36	2:40		
TLIM	000043	1:13	1:23	4:19	4:27			
XLP	000121'	2:44	2:22	2:54				
.1	000121'	3:05	2:16	2:48	3:31	3:46	4:11	
.377	000122'	3:06	1:52					
.60	000116'	3:02	2:33	4:01	4:24			
.CR	000115'	3:01	4:15					
.END	000113'	2:57	1:39					
.IOC1	000134'	3:13	1:30					
.IOC2	000142'	3:21	4:30					
.LF	000114'	2:58	2:52					
.PUT	000107	1:05	3:39					

FLAGGED LINES: NONE

LOCATION LOOKUP (POST.)

The Location Lookup routine reads a location from the text string area of the terminal information table and after some looking through the "XX.STREET.NAMES" and "XX.ADDRESS.MAP" files, it returns a duty post number in the terminal information table. If the lookup fails, it returns a zero.

Input addresses are grouped into three classes:

TYPE 1: < ADDRESS NUMBER > < STREET NAME >
TYPE 2: < STREET NAME > < STREET NAME >
TYPE 3: < PLACE NAME >

Spurious information is ignored.

Examples:

TYPE 1: 725 N. EUCLID
TYPE 2: JACKSON AND RIDGELAND (intersection)
TYPE 3: WHITTIER SCHOOL


```
                .HEAD   A I D S     L O C A T I O N     L O O K U P
                .TITLE  POST.

000107  .PUT=     107     ; FORMATTED OUTPUT
000130  .CCN=     130     ; CONTROL CENTER LOOKUP

000014  FORM=     14      ; FORMAT BLOCK ADDRESS
000016  FIELD=    16      ; FORMAT FIELD NUMBER
000017  INDEX=    17      ; FIELD POSITION INDEX
000020  CCN=      20      ; CONTROL CENTER NAME

177775  FIL=      -3      ; LOCATION FIELD INDEX

000124  TL=       124     ; LOCATION STRING

000041  POST=     P1      ; POST NUMBER

000051  RTRN=     R1      ; MAIN PROGRAM RETURN ADDRESS

000062  BUF=      S2      ; DISK BUFFER HEADER ADDRESS
000063  BSTR=     S3      ; BLOCK STORAGE BASE ADDRESS
000064  CT=       S4      ; LOOP COUNTER
000065  PT=       S5      ; STRING POINTER

                ; BLOCK STORAGE DISPLACEMENTS

000006  SPIN=     6       ; INTERIM STRING ADDRESS
000007  SPEND=    7       ; STRING POINTER LIMIT
000010  ADRV=     10      ; STREET ADDRESS VALUE
000011  COUNT=    11      ; LOOP COUNTER
000012  LBAST=    12      ; LOWER RANGE LIMIT
000013  HOST=     13      ; UPPER RANGE LIMIT
000014  GRAF=     14      ; SECONDARY LOOP COUNTER
000015  SPTR=     15      ; STRING POINTER
000016  STMP=     16      ; TEMPORARY STRING POINTER STORAGE
000017  TPTR=     17      ; TABLE POINTER
000020  TEMP=     20      ; MISC. TEMPORARY STORAGE
000021  SPI=      21      ; SINGLE POINT INDICATOR
000022  SRET=     22      ; SUBROUTINE RETURN ADDRESS
000023  SECR=     23      ; SECONDARY SUBROUTINE RETURN ADDRESS

                .ENT    POST.
                .NREL

000001  .TXTM   1
000001  .TXTX   1
000000'054134   .FN1:   .TXT    *XX.STREET.NAMES*
000010'054134   .FN2:   .TXT    *XX.ADDRESS.MAP*

000020'055051   POST.:  STA 3   RTRN,2  ; SAVE RETURN ADDRESS
000021'132520           SUBZL   0,0     ; CREATE A +1
000022'044050           ENQ             ; ENQUEUE ON THE DISK
000023'126401           SUB     1,1     ; CREATE A ZERO
000024'045041           STA 1   POST,2  ; SET POST = 0
000025'045017           STA 1   INDEX,2 ; SET COLUMN = 0

000026'034565           LDA 3   .30     ; 24 CHARACTERS
```

```
                        STA 3   CT,2     ; SET COUNTER
                        LDA 3   FORM,2   ; POINT AT FORMAT BLOCK
                        LDA 3   T.SA,3   ; START OF FORMAT
                        LDA 1   FIL,3    ; GET OFFSET TO LOCATION FIELD
                        STA 1   FIELD,2  ; POINT TO LOCATION FIELD
                        LDA 3   .TL      ; GET OFFSET
                        ADDZL   2,3      ; CREATE POINTER
                        JMP     .+2      ; HAVE IT ALREADY

LOOP:                   LDA 3   PT,2     ; GET POINTER
                        INC     3,1      ; BUMP IT
                        STA 1   PT,2     ; AND SAVE IT

                        MOVZR   3,3      ; MAKE ADDRESS
                        LDA 1   0,3      ; GET DATA WORD
                        MOVS    1,1 SZC  ; IF EVEN,
                        MOVS    1,1      ; SWAP BYTES

                        JSR     @.PUT    ; OUTPUT CHARACTER
                        DSZ     CT,2     ; DONE?
                        JMP     LOOP     ; NO, REPEAT

                        LDA 0   CCN,2    ; CONTROL CENTER
                        JSR     @.CCN    ; LOCATE IN TABLE
                        JMP     DONE2    ; NOT IN TABLE

                        LDA 1   1,3      ; GET CENTER NAME
                        STA 1   .FN1     ; CREATE 1ST FILE NAME
                        STA 1   .FN2     ; CREATE 2ND FILE NAME

                        SUB     1,1      ; CREATE A ZERO
                        STA 1   INDEX,2  ; POSITION ZERO
                        ISZ     FIELD,2  ; IN NEXT FIELD
                        LDA 1   .18.     ; FIELD LENGTH
                        STA 1   CT,2     ; SET COUNTER
                        LDA 1   .20      ; CITY OFFSET
                        ADDZL   1,3 SKP  ; MAKE BYTE POINTER

DUPE:                   LDA 3   PT,2     ; GET POINTER
                        INC     3,1      ; BUMP IT
                        STA 1   PT,2     ; AND SAVE IT

                        MOVZR   3,3      ; MAKE ADDRESS
                        LDA 1   0,3      ; GET DATA WORD
                        MOVS    1,1 SZC  ; IF EVEN,
                        MOVS    1,1      ; SWAP BYTES

                        JSR     @.PUT    ; OUTPUT IT
                        DSZ     CT,2     ; COUNT IT,
                        JMP     DUPE     ; AND REPEAT

                        LDA 2   .IOCB    ; GET IOCB ADDRESS
                        SUB     0,0      ; CREATE A ZERO
                        STA 0   1,2      ; SET FUNCTION TO ALLOCATE
                        INCZL   0,0      ; CREATE A +2
                        STA 0   2,2      ; SET TYPE TO DISK BUFFER
                        .IO              ; ALLOCATE BUFFER SPACE
DONE2:                  JMP     DONE1    ; QUIT IF NO CAN DO
```



```
                        LDA 2   2,0      ; RESTORE AC2
                        LDA 3   .IOCB    ; GET IOCB ADDRESS
                        LDA 3   2,3      ; GET BUFFER ADDRESS
                        LDA 0   .FND     ; DISPLACEMENT
                        ADD     0,3      ; DISPLACE INTO BUFFER
                        STA 3   BSTR,2   ; SET UP STORAGE ADDRESS
                        LDA 0   .30      ; DISPLACEMENT
                        ADD     0,3      ; DISPLACE INTO BUFFER
                        STA 3   BUF,2    ; SET UP DISK BUFFER ADDRESS

                        LDA 3   .TL      ; GET OFFSET
                        ADDZL   2,3      ; CREATE POINTER
                        LDA 2   BSTR,2   ; SET UP BASE REGISTER
                        STA 3   SPTR,2   ; INITIALIZE STRING POINTER
                        LDA 1   .30      ; LENGTH OF LOCATION STRING
                        ADD     1,3      ; END OF LOCATION STRING
                        STA 3   SPEND,2  ; SAVE IT FOR LATER
                        SUB     3,3      ; CREATE A ZERO
                        STA 3   TPTR,2   ; INITIALIZE TABLE POINTER = 0
                        STA 3   ADRV,2   ; INITIALIZE ADDRESS VALUE = 0
                        JSR     SCAN     ; GET 1ST CHARACTER
                        LDA 1   A0       ; ASCII ZERO
                        LDA 3   A9       ; ASCII NINE
                        SGT     0,3      ; >9?
                        SGE     0,1      ; >=0?
                        JMP     NOTIA    ; NOT A DIGIT

ADRL:                   SUB     1,0      ; CONVERT DIGIT TO BINARY
                        LDA 1   ADRV,2   ; GET ADDRESS VALUE
                        MOVZL   1,3      ; TIMES TWO
                        MOVZL   3,3      ; TIMES FOUR
                        ADDZL   1,3      ; TIMES TEN
                        ADD     0,3      ; COMPUTE NEW ADDRESS VALUE
                        STA 3   ADRV,2   ; SAVE IT

                        ISZ     SPTR,2   ; BUMP STRING POINTER
                        LDA 3   SPTR,2   ; FETCH POINTER
                        MOVZR   3,3      ; CONVERT TO WORD ADDRESS
                        LDA 0   0,3      ; GET DATA WORD
                        MOVS    0,0 SZC  ; IF EVEN,
                        MOVS    0,0      ; SWAP BYTES
                        LDA 1   MASK7    ; FETCH MASK
                        AND     1,1      ; EXTRACT CHARACTER

                        LDA 1   A0       ; ASCII ZERO
                        LDA 3   A9       ; ASCII NINE
                        SGT     0,3      ; >9?
                        SGE     0,1      ; >=0?
                        SKIP             ; NOT A DIGIT
                        JMP     ADRL     ; RECOMPUTE ADDRESS VALUE

                        LDA 1   .40      ; ASCII SPACE
                        SNE     0,1      ; SPACE?
                        JMP     AEND     ; YES, END OF ADDRESS

                        LDA 3   .TL      ; GET OFFSET
                        ADDZL   2,3      ; INITIALIZE POINTER
```

```
                        STA 3   SPTR,2  ; STORE IT
NOT1A:                  JMP     NOT1B   ; ADDRESS IS NOT OF TYPE ONE

AEND:                   JSR     SCAN    ; GET NEXT CHARACTER
                        JSR     SAD     ; TEST FOR A COMPASS POINT
                        105             ; E-AST
                        JMP     MAYBE
                        110             ; N-ORTH
                        JMP     MAYBE
                        123             ; S-OUTH
                        JMP     MAYBE
                        @127            ; W-EST
                        JMP     MAYBE
                        JMP     NOTP1   ; NOT A POINT!

QUIT1:                  JMP     QUIT4   ; LINK

.10:                    10
.20:                    20
.18.:                   18.
.30:                    30
.FW:                    T.FW            ; BUFFER DISPLACEMENT
PTBIT:                  40000           ; POINT REQ'D BIT
.TL:                    TL              ; LOCATION STRING DISPLACEMENT
MASK7:                  177             ; 7-BIT MASK
MASK8:                  377             ; 8-BIT MASK

A0:                     60              ; ASCII ZERO
A9:                     71              ; ASCII NINE

SAD:                    LDA 1   0,3     ; GET COMPARAND
                        SUBOL#  0,1 SNR ; COMPARE
                        JMP     1,3     ; A MATCH
                        INC     3,3     ; MOVE TO NEXT TEST
                        INC     3,3
                        MOVL    1,1 SNC ; IF ANY
                        JMP     SAD     ; CONTINUE
                        JMP     0,3     ; RETURN IF DONE

SCAN:                   STA 3   SRET,2  ; SAVE RETURN ADDRESS
                        LDA 1   .40     ; ASCII SPACE
                        LDA 3   SPTR,2  ; GET STRING POINTER
                        MOVZR   3,3     ; CONVERT TO WORD ADDRESS
                        LDA 0   0,3     ; GET DATA WORD
                        MOVS    0,0 SZC ; IF EVEN,
                        MOVS    0,0     ; SWAP BYTES
                        LDA 3   MASK7   ; FETCH MASK
                        AND     3,0     ; EXTRACT CHARACTER
                        SEQ     0,1     ; SPACE?
                        JMP     @SRET,2 ; NO, RETURN
                        ISZ     SPTR,2  ; YES, BUMP POINTER
                        JMP     SCAN+2  ; TRY THIS CHARACTER

MAYBE:                  STA 0   SPT,2   ; SAVE SINGLE POINT CHARACTER
                        LDA 3   SPTR,2  ; GET STRING POINTER
                        INC     3,3     ; POINT TO NEXT CHARACTER
                        MOVZR   3,3     ; CONVERT TO WORD ADDRESS
                        LDA 0   0,3     ; GET DATA WORD
```


02/20/74

```
 1  00255'101302           MOVS    0,0 SZC ; IF EVEN,
 2  00256'101303           MOVS    0,0     ; SWAP BYTES
 3  00257'024740           LDA 1   MASK7   ; FETCH MASK
 4  00260'123400           AND     1,0     ; EXTRACT CHARACTER
 5
 6  00261'004742           JSR     SAD     ; TEST FOR POINT TERMINATOR
 7  00262'000040  .40:     40              ; SPACE?
 8  00263'000416           JMP     ISP     ; YES, IS A POINT
 9  00264'000056           56".            ; "." ?
10  00265'000414           JMP     ISP     ; YES, IS A POINT
11
12  00266'112400  NOTP1:    SUB     0,0     ; CREATE A ZERO
13  00267'041021           STA 0   SPT,2   ; CLEAR POINT INDICATOR
14  00270'000424           JMP     NOTP    ; NOT A POINT
15
16  00271'000555  NOT1B:    JMP     NOT1    ; LINK
17  00272'000573  DONE1:    JMP     DONE    ; LINK
18
19  00273'000274' .IOCB:    .+1             ; I/O CONTROL BLOCK
20  00274'000000           0               ; CHANNEL
21  00275'000000           0               ; FUNCTION
22  00276'000401           0               ; TYPE / BUFFER ADDRESS
23  00277'000000           0               ; NOT USED
24  00300'000045'          T.FW+30+B.LEN+1 ; SIZE OF BUFFER
25
26  00301'021021  ISP:     LDA 0   SPT,2   ; GET POINT CHARACTER
27  00302'004721           JSR     SAD     ; TEST DIRECTION
28  00303'000105           105             ; E-AST (POSITIVE X)
29  00304'000406           JMP     BUMP    ; ADDRESS IS POSITIVE
30  00305'000116           116             ; N-ORTH (POSITIVE Y)
31  00306'000401           JMP     BUMP    ; ADDRESS IS POSITIVE
32
33  00307'025010           LDA 1   ADRV,2  ; GET ADDRESS VALUE
34  00310'124400           NEG     1,1     ; COMPLEMENT (SOUTH & WEST ARE NEGATIVE
35  00311'045010           STA 1   ADRV,2  ; RESTORE ADDRESS VALUE
36
37  00312'011015  BUMP:    ISZ     SPTR,2  ; POINT PAST COMPASS POINT,
38  00313'011015           ISZ     SPTR,2  ; AND ITS TERMINATOR
39
40  00314'004562  NOTP:    JSR     FINDA   ; EXTRACT A STREET NAME
41  00315'035017           LDA 3   TPTR,2  ; GET TABLE POINTER
42  00316'175405           MOV     3,3 SNR ; ZERO?
43  00317'000450           JMP     QUIT4   ; YES, NO MATCH FOUND
44
45  00320'021003           LDA 0   0,2     ; GET HEADER WORD
46  00321'105102           MOVL    0,1 SZC ; A STREET ENTRY?
47  00322'000531           JMP     PLACE   ; NO, A PLACE NAME
48
49  00323'024675  LNKUP:    LDA 1   MASK8   ; LOW BYTE MASK
50  00324'124400           AND     0,1     ; EXTRACT NUMBER OF ENTRIES
51  00325'045411           STA 1   COUNT,2 ; STORE IN COUNT
52
53  00326'021667           LDA 1   PTBIT   ; GET POINT BIT MASK
54  00327'174404           AND     0,1     ; GET POINT REQ'D BIT
55  00330'021021           LDA 0   SPT,2   ; AND POINT INDICATOR
56  00331'101004           MOV     0,0 SZR ; IS THERE A POINT?
57  00332'000403           JMP     .+3     ; YES, USE IT!
58  00333'125004           MOV     1,1 SZR ; NO, DO I NEED ONE?
```

```
                    JMP     QUIT4   ; YES, TOO BAD

                    JSR     SAD     ; TEST FOR A NORTH-SOUTH STREET
                    "E
                    JMP     LOOP1   ; EAST IS OK
                    0"N
                    JMP     LOOP1   ; SO IS WEST
                    NEG     0,0
                    STA 0   SPI,2   ; MINUS SPI ON NORTH-SOUTH STREETS

LOOP1:              JSR     FILL    ; INPUT A RECORD
                    INC     2,3     ; MOVE TO BEGINNING OF SEGMENT

                    LDA 0   0,3     ; GET BLOCK WORD
                    MOVL    0,1     ; MOVE SIGN TO CARRY
                    LDA 1   SPI,2   ; GET DIRECTION
                    MOVR    0,0 SNC ; TEST DIRECTION BIT
                    JMP     TEW     ; TEST EAST WEST
                    MOV     1,1 SNR ; POINT GIVEN?
                    JMP     DMCH    ; NONE MATCHES ANYTHING
                    MOVL    1,1 SZC ; N OR S ADDR?
                    JMP     DMCH    ; YES, MATCHES SEGMENT
                    JMP     TNXT    ; NO, TRY NEXT SEGMENT

FILL:               STA 3   SRET,2  ; SAVE RETURN
                    LDA 0   .10     ; 8 BYTE SEGMENTS
                    INCZL   2,1     ; POINT PAST HEADER
                    LDA 3   DDOS    ; GET SYSTEM TABLE ADDRESS
                    LDA 2   2,0     ; RESTORE BASE REGISTER
                    LDA 2   BUF,2   ; BUFFER PREFIX ADDRESS
                    JSR     @D.WIN,3        ; READ IN DATA BLOCK
QUIT4:              JMP     QUIT    ; QUIT ON EOF

                    LDA 2   2,0     ; RESTORE BASE REGISTER
                    LDA 2   BSTR,2  ; SET NEW BASE REGISTER
                    JMP     @SRET,2 ; RETURN

TEW:                MOV     1,1 SNR ; POINT GIVEN?
                    JMP     DMCH    ; NONE MATCHES ANYTHING
                    MOVL    1,1 SZC ; E OR W ADDR?
                    JMP     TNXT    ; NO, TRY NEXT SECMENT

DMCH:               STA 0   0,3     ; TRUE BLOCK VALUE
                    LDA 0   1,3     ; GET LEAST
                    MOVR    0,0 SNC ; ONLY ONE SIDE OF STREET?
                    JMP     EASY    ; EASY CASE, BOTH SIDES

                    LDA 0   2,3     ; GET MOST
                    LDA 1   ADRV,2  ; GET ADDRESS VALUE
                    ADDZR   0,1 SZC ; PROPER SIDE OF STREET?
                    JMP     TNXT    ; NO, TRY NEXT SEGMENT

EASY:               LDA 0   2,3     ; GET MOST
                    MOVL    0,1     ; MOVE SIGN TO CARRY
                    MOVR    0,0     ; REMOVE FLAG BIT
                    STA 0   MOST,2  ; SAVE TRUE MOST VALUE

                    LDA 0   1,3     ; GET LEAST
```



```
                    MOVL    0,1     ; MOVE SIGN TO CARRY
                    MOVR    0,0     ; REMOVE FLAG BIT
                    STA 0   LEAST,2 ; SAVE TRUE LEAST VALUE

                    LDA 1   ADRV,2  ; GET ADDRESS VALUE
                    SGE     1,0     ; >= LEAST ?
                    JMP     ZSPI    ; NO, CHECK FOR SPI=0

                    LDA 0   MOST,2  ; GET MOST VALUE
                    SLE     1,0     ; <= MOST ?
                    JMP     ZSPI    ; NO CHECK FOR SPI=0

GOTIT:              LDA 2   2,0     ; SET BASE REGISTER
                    LDA 0   3,3     ; GET POST
                    STA 0   POST,2  ; RETURN POST
                    JMP     QUIT    ; DEALLOCATE BUFFERS

ZSPI:               LDA 0   SPI,2   ; GET SINGLE POINT INDICATOR
                    MOV     0,0 SZR ; WAS THERE A POINT?
                    JMP     TNXT    ; YES, AND IT FAILED
                    NEG     1,1     ; NO, TRY COMPLEMENT

                    LDA 0   LEAST,2 ; GET LEAST VALUE
                    SGE     1,0     ; >= LEAST?
                    JMP     TNXT    ; NO, TRY NEXT SEGMENT

                    LDA 0   MOST,2  ; GET MOST VALUE
                    SGT     1,0     ; <= MOST?
                    JMP     GOTIT   ; YES, WE MADE IT!

TNXT:               DSZ     COUNT,2 ; MORE TO TRY?
                    JMP     LOOP1   ; YES, CONTINUE
                    JMP     QUIT    ; NOPE, TOO BAD

NOT1:               JSR     FIND    ; FIND A NAME
                    LDA 3   TPTR,2  ; GET TABLE POINTER
                    MOV     3,3 SNR ; NULL?
                    JMP     QUIT    ; YES, TOO BAD

                    LDA 0   0,2     ; GET HEADER WORD
PLACE:              MOVZL   0,0 SNC ; TABLE ENTRY A PLACE?
                    JMP     TYPE2   ; NO, ADDRESS IS OF THE SECOND TYPE

                    LDA 2   2,0     ; SET BASE REGISTER
                    MOVZR   0,0     ; REMOVE TOP BIT
                    STA 0   POST,2  ; RETURN POST

QUIT:               LDA 2   .IOCB   ; GET IOCB ADDRESS
                    SUBZL   0,0     ; CREATE A +1
                    STA 0   1,2     ; SET FUNCTION TO DEALLOCATE
                    .IO             ; DEALLOCATE DISK BUFFER
                    ABORT

DONE:               LDA 2   2,0     ; RESTORE BASE REGISTER
                    SUBZL   0,0     ; CREATE A +1
                    DEQ             ; DEQUEUE ON THE DISK
                    JMP     @RTRN,2 ; RETURN
```

```
                .N:     "N              ; ASCII "N"
                .E:     "E              ; ASCII "E"
                .W:     "W              ; ASCII "W"
                .S:     "S              ; ASCII "S"

                LUKUP:  JMP     LNKUP   ; LINK
                FINDA:  JMP     FIND    ; LINK

                ..0:    "0              ; ASCII ZERO
                ..9:    "9              ; ASCII NINE

                TYPE2:  JSR     FILL    ; INPUT A SEGMENT
                        LDA 0   1,2     ; GET BLOCK WORD
                        MOVL    0,1     ; SET CARRY TO SIGN
                        MOVR    0,0 SZC ; REMOVE DIRECTION BIT
                        JMP     EWS     ; EAST - WEST POINT

                        SUB     1,1     ; CREATE A ZERO
                        SGE     0,1     ; NORTH POINT?
                        JMP     .+3     ; NO, SOUTH
                        LDA 1   .N      ; ASCII "N"-ORTH
                        JMP     SPOK    ; SET POINT
                        LDA 1   .S      ; ASCII "S"-OUTH
                        JMP     SPOK    ; SET POINT

                EWS:    SUB     1,1     ; CREATE A ZERO
                        SGE     0,1     ; EAST POINT?
                        JMP     .+3     ; NO, WEST
                        LDA 1   .E      ; ASCII "E"-AST
                        JMP     SPOK    ; SET POINT
                        LDA 1   .W      ; ASCII "W"-EST

                SPOK:   STA 1   SPI,2   ; SET SINGLE POINT INDICATOR
                        STA 0   ADRV,2  ; SET ADDRESS VALUE TO BLOCK
                        LDA 3   SPIN,2  ; GET INTERIM STRING POINTER
                        STA 3   SPTR,2  ; RESET STRING POINTER

                SPX:    JSR     SCHR    ; GET NEXT STRING CHARACTER
                        LDA 1   ..0     ; ASCII ZERO
                        SGE     0,1     ; ALPHA?
                        JMP     .-3     ; NO, TRY NEXT ONE

                        DSZ     SPTR,2  ; POINT BACK TO THIS CHARACTER
                        LDA 3   SPEND,2 ; GET END OF STRING
                        LDA 1   SPTR,2  ; GET STRING POINTER
                        SLT     1,3     ; STRING EXHAUSTED?
                        JMP     QUIT    ; YES

                        SUB     1,1     ; CREATE A ZERO
                        STA 1   TPTR,2  ; SET TABLE POINTER TO ZERO
                        JSR     FIND    ; FIND A NAME
                        LDA 0   TPTR,2  ; GET TABLE POINTER
                        MOV     0,0 SNR ; TABLE POINTER ZERO?
                        JMP     MOVE    ; YES, MOVE STRING POINTER

                        LDA 0   0,2     ; GET HEADER WORD
                        MOVL    0,0 SNC ; TABLE ENTRY A PLACE?
                        JMP     LUKUP   ; NO, DO LOOK UP
```



```
                MOVE:   LDA 3   STMP,2  ; GET INITIAL STRING POINTER
                        STA 3   SPTR,2  ; SET STRING POINTER
                        JSR     SCHR    ; GET NEXT STRING CHARACTER
                        LDA 1   ..0     ; ASCII ZERO
                        SLT     0,1     ; ALPHA?
                        JMP     .-3     ; YES, TRY NEXT ONE
                        JMP     SPX     ; TEST FOR EXHAUSTION

                SCHR:   STA 3   SECR,2  ; SAVE RETURN
                        LDA 3   SPTR,2  ; GET STRING POINTER
                        MOVZR   3,3     ; CONVERT TO WORD ADDRESS
                        LDA 0   0,3     ; GET DATA WORD
                        MOVS    0,0 SZC ; IF EVEN,
                        MOVS    0,0     ; SWAP BYTES
                        LDA 1   .177    ; FETCH MASK
                        AND     1,0     ; EXTRACT CHARACTER
                        ISZ     SPTR,2  ; BUMP STRING POINTER
                        JMP     @SECR,2 ; RETURN

                FIND:   STA 3   SRET,2  ; SAVE RETURN ADDRESS
                        LDA 3   SPTR,2  ; GET INITIAL VALUE OF STRING POINTER
                        STA 3   STMP,2  ; SAVE IT

                        LDA 1   FNI     ; GET FILE NAME POINTER
                        JSR     PRIME   ; PRIME INPUT BUFFER

                FIND1:  JSR     REED    ; READ A FILE CHARACTER
                        STA 0   TEMP,2  ; SAVE IT
                        JSR     SCHR    ; GET NEXT STRING CHARACTER
                        LDA 1   ..0     ; ASCII ZERO
                        SGE     0,1     ; ALPHA?
                        JMP     .-3     ; YES, TRY NEXT ONE

                        LDA 1   TEMP,2  ; PICK UP FILE CHARACTER
                        SEQ     0,1     ; SAME?
                        JMP     SPEW    ; NO, TRY NEXT FILE ENTRY

                FNXT:   JSR     REED    ; READ A FILE CHARACTER
                        STA 0   TEMP,2  ; SAVE IT
                        JSR     SCHR    ; GET NEXT STRING CHARACTER
                        LDA 1   ..0     ; ASCII ZERO
                        SGE     0,1     ; ALPHA?
                        LDA 0   .SP     ; NO, MAKE SPACE

                ENXT:   LDA 1   TEMP,2  ; AND FILE CHARACTER
                        SEQ     0,1     ; SAME?
                        JMP     SPEW    ; NO, TRY NEXT FILE ENTRY

                        LDA 1   .SP     ; ASCII SPACE
                        SEQ     0,1     ; SPACE?
                        JMP     FNXT    ; NO, COMPARE NEXT PAIR

                        JSR     REED    ; READ NEXT FILE CHARACTER
                        STA 0   TEMP,2  ; SAVE IT
                        LDA 1   .STAR   ; ASCII "*"
                        SNE     0,1     ; STAR?
                        JMP     ISTR    ; YES, CONTINUE
```

```
                        JSR     SCHR    ; GET NEXT STRING CHARACTER
                        LDA 1   "0      ; ASCII ZERO
                        SGE     0,1     ; ALPHA?
                        JMP     .-3     ; NO, TRY NEXT ONE
                        JMP     ENXT    ; COMPARE NOW

                QUIT3:  JMP     QUIT    ; LINK

                ISTR:   SUB     3,3     ; CREATE A ZERO
                        STA 3   TPTR,2  ; SET TABLE POINTER = 0

                        JSR     REED    ; READ NEXT FILE CHARACTER
                        LDA 1   "0      ; ASCII ZERO
                        LDA 3   "9      ; ASCII NINE
                        SGT     0,3     ; >9 ?
                        SGE     0,1     ; >=0 ?
                        JMP     NOTD    ; NOT A DIGIT

                        SUB     1,0     ; CONVERT TO BINARY
                        LDA 3   TPTR,2  ; GET TABLE POINTER
                        MOVZL   3,1     ; TIMES TWO
                        MOVZL   1,1     ; TIMES FOUR
                        ADDZL   1,3     ; TIMES TEN
                        ADD     0,3     ; COMPUTE NEW POINTER
                        STA 3   TPTR,2  ; SAVE IT
                        JMP     ISTR+2  ; GET NEXT CHARACTER

                NOTD:   LDA 1   .12     ; ASCII LINE FEED
                        SEQ     0,1     ; END OF LINE?
                        JMP     ISTR+2  ; NO, TRY NEXT CHARACTER
                        LDA 3   SPTR,2  ; FETCH STRING POINTER
                        STA 3   SPIN,2  ; SAVE IT FOR NEXT PASS
                        JMP     RSET    ; YES, RESET STRING POINTER

                SPEW:   LDA 1   TPTR,2  ; GET TABLE POINTER
                        MOV     1,1 SZR ; ANY LUCK?
                        JMP     EXIT    ; YES, GOT SOMETHING
                        JSR     REED    ; READ NEXT FILE CHARACTER
                        LDA 1   .12     ; ASCII LINE FEED
                        SEQ     0,1     ; END OF LINE?
                        JMP     .-3     ; NO, TRY NEXT CHARACTER

                RSET:   LDA 3   STMP,2  ; GET INITIAL STRING POINTER
                        STA 3   SPTR,2  ; RESET STRING POINTER
                        JMP     FIND1   ; TRY NEXT FILE ENTRY

                REED:   STA 3   SECR,2  ; SAVE RETURN ADDRESS
                        LDA 3   DDOS    ; GET SYSTEM ADDRESS
                        LDA 2   2,0     ; RESTORE BASE REGISTER
                        LDA 2   BUF,2   ; BUFFER PREFIX ADDRESS
                        JSR     @D.IN,3 ; READ A BYTE
                        JMP     EXIT    ; MAIN RETURN ON EOF

                        LDA 1   .177    ; FETCH MASK
                        AND     1,0     ; EXTRACT CHARACTER
                        LDA 2   2,0     ; RESTORE BASE REGISTER
                        LDA 2   BSTR,2  ; SET NEW BASE REGISTER
```



```
00711 003023            JMP     @SECR,2 ; RETURN
00712 000012    .12:    12              ; ASCII LINE FEED
00713 000052    .STAR:  "*              ; ASCII STAR "*"
00714 000040    .SP:    40              ; ASCII SPACE
00715 000177    .177:   177             ; 7-BIT MASK
00716 000377    .377:   377             ; HALF WORD MASK
00717 000000    FN1:    .FN1            ; FIRST FILE NAME
00720 000000    FN2:    .FN2            ; SECOND FILE NAME

00721 055023    PRIME:  STA 3   SECR,2  ; SAVE RETURN ADDRESS
00722 034041            LDA 3   DDOS    ; GET SYSTEM ADDRESS
00723 131000            MOV     1,2     ; POINT TO FILE NAME
00724 004402            JSR     @D.SRC,3        ; SEARCH FOR FILE
00725 000711    QUIT2:  JMP     QUIT3   ; FILE NOT FOUND

00726 021010            LDA 0   F.ID,2  ; GET FILE IDENTIFIER
00727 041454            STA 0   D.SP3,3 ; SET UP PARMS FOR PRIME
00730 021013            LDA 0   F.FST,2 ; HIGH ORDER DISK ADDRESS
00731 025015            LDA 1   F.BFP,2 ; LOW ORDER DISK ADDRESS
00732 031764            LDA 2   .377    ; HALF WORD MASK
00733 101221            MOVZR   0,0     ; SHIFT HIGH ORDER ADR, THEN
00734 147500            ANDL    2,1     ; LOW ORDER, WITH CARRY
00735 030002            LDA 2   2,0     ; SET UP BASE REGISTER
00736 031062            LDA 2   BUF,2   ; BUFFER PREFIX ADDRESS
00737 005753            JSR     D.PRM,3 ; PRIME INPUT BUFFER

00740 030002            LDA 2   2,0     ; RESTORE BASE REGISTER
00741 031063            LDA 2   BSTR,2  ; SET NEW BASE REGISTER
00742 003023            JMP     @SECR,2 ; RETURN

00743 030002    EXIT:   LDA 2   2,0     ; RESET BASE REGISTER
00744 031063            LDA 2   BSTR,2  ; SET NEW BASE REGISTER
00745 025017            LDA 1   TPTR,2  ; GET TABLE POINTER
00746 125005            MOV     1,1 SNR ; FIND ANYTHING?
00747 003022            JMP     @SRET,2 ; NOPE, RETURN
00750 045011            STA 1   COUNT,2 ; SET UP COUNTER
00751 024747            LDA 1   FN2     ; GET FILE NAME POINTER
00752 004747            JSR     PRIME   ; PRIME INPUT BUFFER

00753 015011    FLUP:   DSZ     COUNT,2 ; BUMP COUNTER
00754 000402            SKIP            ; NOT DONE, CONTINUE
00755 000414            JMP     FOUT    ; DONE SPEWING

00756 004416            JSR     READ2   ; GET HEADER
00757 141112            MOVL#   0,0 SZC ; PLACE? (I.E. NOT A STREET)
00760 000773            JMP     FLUP    ; YES, ONLY HEADER
00761 024735            LDA 1   .377    ; HALF WORD MASK
00762 123003            AND     1,0     ; EXTRACT # OF SEGMENTS
00763 103120            ADDZL   0,0     ; 4 WDS / SEGMENT
00764 041014            STA 0   GRAF,2  ; SET UP COUNTER
00765 004407            JSR     READ2   ; SPEW A FULL WORD
00766 015014            DSZ     GRAF,2  ; DONE?
00767 000776            JMP     .-2     ; NO, SPEW AGAIN
00770 000763            JMP     FLUP    ; DONE WITH THIS ENTRY

00771 004403    FOUT:   JSR     READ2   ; READ HEADER
00772 041000            STA 0   0,2     ; SAVE IT
```

```
                    JMP     @SRET,2 ; YES, DONE

                READ2:  STA 3   SECR,2  ; SAVE RETURN ADDRESS
                        LDA 3   DDOS    ; GET SYSTEM ADDRESS
                        LDA 2   2,0     ; RESTORE BASE REGISTER
                        LDA 2   BUF,2   ; BUFFER PREFIX ADDRESS
                        JSR     @D.IN,3 ; READ IN HIGH BYTE
                        JMP     QUIT2   ; QUIT ON EOF

                        LDA 2   2,0     ; RESTORE BASE REGISTER
                        LDA 2   BSTR,2  ; SET NEW BASE
                        STA 0   TEMP,2  ; SAVE HIGH BYTE

                        LDA 2   2,0     ; RESTORE BASE REGISTER
                        LDA 3   DDOS    ; GET SYSTEM ADDRESS
                        LDA 2   BUF,2   ; BUFFER PREFIX ADDRESS
                        JSR     @D.IN,3 ; READ IN LOW BYTE
                        JMP     QUIT2   ; QUIT ON EOF

                        LDA 2   2,0     ; RESTORE BASE REGISTER
                        LDA 2   BSTR,2  ; SET NEW BASE
                        LDA 1   TEMP,2  ; PICK UP HIGH BYTE
                        MOVS    1,1     ; MOVE TO UPPER BYTE
                        ADD     1,0     ; ADD IN LOW BYTE
                        JMP     @SECR,2 ; RETURN

                .END
```


SYMBOL	VALUE	DEF'N	REFERENCES						
A0	000221'	4:28	3:23	3:46					
A9	000222'	4:29	3:24	3:47					
ADRL	000140'	3:29	3:51						
ADRV	000010	1:30	3:21	3:30	3:35	5:33	5:35	6:49	7:05
			8:34						
AEND	000174'	4:04	3:55						
BSTR	000063	1:22	3:07	3:14	6:35	10:58	11:29	11:33	12:11
			12:21						
BUF	000062	1:21	3:10	6:30	10:51	11:25	12:06	12:16	
BUMP	000312'	5:37	5:29	5:31					
CCN	000020	1:11	2:23						
COUNT	000011	1:31	5:51	7:31	11:37	11:41			
CT	000064	1:23	2:01	2:20	2:35	2:49			
DBGCT	000377'	6:43	6:20	6:22	6:39				
DONE	000465'	7:54	5:17						
DONE1	000272'	5:17	2:53						
DONE2	000106'	2:53	2:25						
DUPE	000066'	2:39	2:54						
EASY	000407'	6:53	6:46						
EXIT	000616'	9:46	10:06						
FMS	000515'	8:26	8:16						
FIELD	000016	1:09	2:05	2:33					
FIL	177775	1:13	2:34						
FILL	000364'	6:25	6:11	8:12					
FIND	000572'	9:21	7:35	8:07	8:51				
FI	000577'	9:28	10:46						
FINDA	000476'	8:07	5:43						
FLUP	000763'	11:41	11:47	11:55					
FN1	000717'	11:08	9:25						
FN2	000723'	11:09	11:38						
FNXT	000610'	9:39	9:52						
FORM	000014	1:08	2:02						
FOUT	000771'	11:57	11:43						
FXIT	000743'	11:32	10:33	10:53					
GOTIT	000425'	7:13	7:29						
GNAF	000014	1:34	11:51	11:53					
INDEX	000017	1:10	1:56	2:32					
ISP	000301'	5:26	5:08	5:10					
ISTR	000637'	10:10	9:53	10:27	10:31				
LEAST	000012	1:32	7:03	7:23					
LNKUP	000323'	5:49	8:46						
LOOP	000037'	2:10	2:21						
LOOP1	000344'	6:11	6:05	6:07	7:32				
LUKUP	000475'	8:06	8:53						
MASK7	000217'	4:25	3:43	4:47	5:03				
MASK8	000220'	4:26	5:49						
MAYBE	000254'	4:54	4:07	4:09	4:11	4:13			
MODT	000013	1:33	6:56	7:09	7:27				
MOVE	000551'	9:02	8:54						
NUM1	000345'	7:35	5:16						
NO[illegible]	000173'	4:02	3:27						
NOT0	000271'	5:16	4:02						
NOT0	000657'	10:29	10:13						
NTP	000314'	5:40	5:14						
NOTP1	000266'	5:12	4:14						

SYMBOL	VALUE	DEF'N	REFERENCES						
	[illegible]453'	7:41	5:47						
	[illegible]41	1:17	1:55	7:15	7:46				
	[illegible]20'	1:51	1:43						
	[illegible]721'	11:11	9:26	11:39					
	[illegible]5	1:24	2:10	2:12	2:39	2:41			
	[illegible]216'	4:23	5:53						
	[illegible]4	7:43	6:32	7:16	7:33	7:38	8:47	10:08	
	[illegible]7'	4:16							
	[illegible]725'	11:15	12:03	12:13					
	[illegible]35'	11:03	11:15						
	[illegible]47'	6:32	4:16	5:43	6:01				
	[illegible]34'	12:13	11:45	11:52	11:57				
	[illegible]77'	10:46	9:28	9:39	9:54	10:13	10:39		
	[illegible]71'	10:44	10:34						
	[illegible]1	1:19	1:51	7:57					
	[illegible]23'	4:31	4:05	4:37	5:06	5:27	6:03		
	[illegible]33'	4:31	3:22	4:04	4:52				
	[illegible]4'	9:14	8:33	9:04	9:30	9:41	10:02		
	[illegible]3	1:41	9:10	9:19	10:48	11:01	11:11	11:30	12:03
			12:25						
	[illegible]007	1:29	3:18	8:44					
	[illegible]03'	1:36	9:37	9:48					
	[illegible]21	1:39	4:54	5:13	5:26	5:55	6:09	6:16	7:18
			8:33						
	[illegible]6	1:28	8:35	10:33					
	[illegible]523'	8:33	8:22	8:24	8:30				
	[illegible]815	1:35	3:15	3:37	3:38	4:01	4:42	4:51	4:55
			5:37	5:38	8:36	8:43	8:45	9:03	9:11
			9:18	9:22	10:32	10:45			
	[illegible]27'	8:38	9:03						
	[illegible]92	1:40	4:49	4:50	6:25	6:36	9:21	11:36	12:01
	[illegible]16	1:36	9:02	9:23	10:44				
	[illegible]820	1:38	9:29	9:35	9:40	9:46	9:55	12:12	12:22
	[illegible]373'	6:38	6:18						
	[illegible]1	1:15	4:24						
	[illegible]43'	7:31	6:23	6:41	6:51	7:20	7:25		
	[illegible]7	1:37	3:20	5:41	7:36	8:50	8:52	10:11	10:21
			10:26	10:36	11:34				
	[illegible]11'	4:12	7:42						
	[illegible]431'	7:18	7:07	7:11					
	[illegible]21'	4:18	6:26						
	[illegible]712'	11:03	10:29	10:44					
	[illegible]715'	11:06	9:16	10:55					
	[illegible]212'	4:20	2:34						
	[illegible]11'	4:19	2:36						
	[illegible]13'	4:21	1:58	3:08	3:16				
	[illegible]716'	11:07	11:21	11:48					
	[illegible]2'	5:47	3:53	4:41					
	[illegible]31'	1:36	2:24						
	[illegible]7'	4:32	8:29						
	[illegible]'	1:48	2:23	11:08					
	[illegible]10'	1:49	2:29	11:09					
	[illegible]214'	4:22	3:05						
	[illegible]'	5:17	2:52	3:03	7:48				
	[illegible]71'	8:31	8:21						

SYMBOL	VALUE	DEF'N	REFERENCES					
.PUT	000107	1:05	2:19	2:48				
.S	000474'	8:04	8:23					
.SP	000714'	11:05	9:44	9:50				
.STAR	000713'	11:04	9:56					
.TL	000216'	4:24	2:06	3:12	3:57			
..I	000473'	8:03	8:31					
..0	000477'	8:09	8:39	9:05	9:31	9:42	10:03	10:14
..9	000504'	8:10	10:15					

FLAGGED LINES: NONE

TICKET FUNCTION (FIND.)

The ticket function types "TICKET" on the display and waits for a ticket number to be typed in. If the input ticket number is zero, the most recently displayed ticket is made current. If the requested ticket is already present in core, its data block use count is incremented. It it is not in core, it is added to the block list and read in from disk. In any case, the ticket is displayed on the terminal using .DUMP.

The ticket function plugs its internal address "STAT" into the end control word. The section of code at "STAT" sets up for and makes a call to the ticket status output routine ".TSO". If instead of typing in a ticket number the operator strikes the status key, the status function will display the ticket status table.

Several routines make use of portions of this function by passing a ticket number in AC1.


```
                        .HEAD   A I D S    T I C K E T    F U N C T I O N
                        .TITLE  FIND.
                        ;       AIDS TABLE ASSIGNMENTS
                        .EXTD   TYPE    ; TEXT STRING TYPER
                        .EXTD   ASK     ; DATA STRING INPUT
                        .EXTD   .CVA    ; ASCII CONVERSION
                        .EXTD   .CVB    ; BINARY CONVERSION
                        .EXTD   .TSO    ; TICKET STATUS OUTPUT
                        .EXTD   .DUMP   ; FORMATTED BLOCK OUTPUT
                        .EXTD   .FORM   ; TICKET FORMAT BLOCK
                        .EXTD   LIST    ; BLOCK LIST POINTER
                        .EXTD   .KILL   ; DELETE CODE BLOCK
                        .EXTD   .TOP    ; ENTER EDIT MODE
                        .EXTD   TAD     ; TAG-LINE ADDRESS
                        ;       TIT WORD ASSIGNMENTS
                        .EXTD   ZAP     ; DISPLAY COMMAND ENTRY
                        .EXTD   LINK    ; GENERAL LINK WORD
                        .EXTD   DATA    ; TICKET BLOCK ADDRESS
                        .EXTD   FORM    ; FORMAT BLOCK ADDRESS
                        .EXTD   CCN     ; CONTROL CENTER NAME
                        .EXTD   TKT     ; CURRENT TICKET NUMBER
                        .EXTD   OLD     ; PREVIOUS TICKET NUMBER
                        .EXTD   END     ; END CONTROL WORD
        000043  D5=     P3
        000044  D4=     P4
        000045  D3=     P5      ; DECIMAL DIGITS
        000046  D2=     P6
        000047  D1=     P7
        000050  RC=     R0      ; RETURN ADDRESS
        000051  RA=     R1      ; ALTERNATE RETURN
        000062  LAST=   S2      ; LAST TICKET (FOR STATUS)
        000100  FN=     100     ; FILE NAME
        000110  TN=     110     ; INPUT DATA
        000100  F1=     FN
        000101  F2=     FN+1    ; FILE NAME
        000102  F3=     FN+2
                        .ENT    FIND.,FALT.
                        .NREL
00000'055050  FIND.:    STA 3   RC,2    ; SAVE RETURN
00001'006011$           JSR     @.KILL  ; DELETE CODE
00002'034474            LDA 3   STAT    ; STATUS ENTRY
00003'055023$           STA 3   END,2   ; INTO THE TIT
00004'102400            SUB     0,0
00005'041062            STA 0   LAST,2  ; FOR STATUS
```

```
                ; REQUEST TICKET NUMBER

                        SUB     0,0
                        LDA 1   TAD     ; POSITION
                        JSR     @ZAP,2

                        JSR     @TYPE   ; TYPE "TICKET"
                        MSG
                        LDA 0   .5
                        LDA 1   .TN     ; STRING OFFSET
                        ADD     2,1     ; MAKE ADDRESS
                        JSR     @ASK    ; GET NUMBER
                        JMP     @RC,2   ; IF NONE, EXIT

                ; COMPUTE TICKET NUMBER

                        LDA 0   TN,2
                        STA 0   D4,2    ; DIGIT 4
                        MOVS    0,0
                        STA 0   D5,2    ; AND 5

                        LDA 0   TN+1,2
                        STA 0   D2,2    ; DIGIT 2
                        MOVS    0,0
                        STA 0   D3,2    ; AND 3

                        LDA 0   TN+2,2
                        MOVS    0,0     ; DIGIT 1
                        STA 0   D1,2

                        JSR     @.CVB   ; MAKE BINARY
                        JSR     FALT.   ; FIND THE TICKET
                        JSR     @.DUMP  ; AND DISPLAY IT
                        LDA 3   RC,2    ; RETURN ADDRESS
                        JMP     @.TOP   ; EXIT THRU TOP
```



```
                ; ALTERNATE ENTRY

000040 055051   FALT.:  STA 3   RA,2     ; SAVE RETURN
000041 125015           MOV#    1,1 SNR ; IF TICKET ZERO,
000042 025022$          LDA 1   OLD,2   ; SUBSTITUTE
000043 125015           MOV#    1,1 SNR ; IF STILL ZERO,
000044 000455           JMP     ERROR   ; COMPLAIN

000045 035021$          LDA 3   TKT,2   ; CURRENT TICKET
000046 136414           SEQ     1,3     ; IF DIFFERENT
000047 055022$          STA 3   OLD,2   ; BECOMES OLD ONE

000050 035010$          LDA 3   DATA,2  ; TICKET BLOCK
000051 175414           MOV#    3,3 SZR ; IF ANY DATA,
000052 015425$          DSZ     USE,3   ; REDUCE COUNT
000053 000403           JMP     GET     ; CONTINUE

000054 054016$          STA 3   LIST    ; NEXT IN LINE
000055 006440           IDLE            ; IN CASE OF SKIP

                ; CONSTRUCT FILE NAME

000056 015021$  GET:    STA 1   TKT,2   ; TICKET NUMBER
000057 060043$          JSR     @.CVA   ; MAKE ASCII

000060 021043           LDA 0   D5,2    ; DIGIT FIVE
000061 101304           MOVS    0,0     ; TO TOP BYTE
000062 025041           LDA 1   D4,2    ; DIGIT FOUR
000063 123003           ADD     1,0     ; COMBINE
000064 041100           STA 0   F1,2    ; AND SAVE

000065 021045           LDA 0   D3,2    ; DIGIT THREE
000066 101301           MOVS    0,0     ; TO TOP BYTE
000067 025046           LDA 1   D2,2    ; DIGIT TWO
000070 123000           ADD     1,0     ; COMBINE
000071 041101           STA 0   F2,2    ; AND SAVE

000072 021047           LDA 0   D1,2    ; DIGIT ONE
000073 101303           MOVS    0,0     ; TO TOP BYTE
000074 041102           STA 0   F3,2    ; AND SAVE
000075 000457           JMP     ALL
```

```
                        ; DISPLAY STATUS

                STAT:   .+1
                        STA 3   RC,2    ; SAVE RETURN

                        LDA Ø   TN,2
                        STA Ø   D4,2    ; DIGIT 4
                        MOVS    Ø,Ø
                        STA Ø   D5,2    ; AND 5

                        LDA Ø   TN+1,2
                        STA Ø   D2,2    ; DIGIT 2
                        MOVS    Ø,Ø
                        STA Ø   D3,2    ; AND 3

                        LDA Ø   TN+2,2
                        MOVS    Ø,Ø     ; DIGIT 1
                        STA Ø   D1,2

                        JSR     @.CVB   ; MAKE BINARY
                        LDA Ø   TN,2    ; INPUT DATA
                        MOV#    Ø,Ø SNR ; IF NONE,

                        LDA Ø   CCN,2   ; CONTROL CENTER
                        LDA 3   RC,2    ; RETURN ADDRESS
                        JMP     @.TSØ   ; STATUS OUTPUT

                        ; TYPE ERROR MESSAGE

                ERROR:  LDA 1   ERR     ; TYPE "?"
                        LDA Ø   .13Ø
                        LDA 3   RC,2    ; RETURN ADDRESS
                        JMP     @ZAP,2  ; EXIT THRU ZAP
```

```
00125  000005  .5:     5
00126  000010  .FWD:   T.FW
00127  000130  .13Ø:   13Ø
00130  000377  .377:   377

00131  000110  .TN:    TN      ; TICKET OFFSET
00132  000100  .FN:    FN      ; FILE-NAME OFFSET

00133  000134  .IOC1:  .+1     ; ALLOCATE IOCB
00134  000041          Ø
00135  000004          Ø
00136  000000  BLK1:   Ø
00137  000000          Ø
00140  000424          T.FW+B.LEN+1

00141  000142  .IOC2:  .+1     ; DEALLOCATE IOCB
00142  000000          Ø
00143  000001          1
00144  000000  BLK2:   Ø

00145  000000  SAVE:   Ø       ; ADDRESS SAVE

       000001  .TXTM   1
       000001  .TXTX   1

00146  003477  ERR:    7*400+"?
00147  052111  MSG:    .TXT    *TICKET*

                ; TICKET DATA BLOCK ASSIGNMENTS

                .EXTD   MOD     ; DATA BLOCK UPDATE FLAG
                .EXTD   USE     ; DATA BLOCK USAGE COUNT
                .EXTD   REF     ; TICKET NUMBER IN DATA
                .EXTD   BLK     ; BLOCK NUMBER IN TICKET
```

```
                        ; SEARCH BLOCK LIST

                        IDLE
                ALL:    LDA 3   LIST    ; LIST POINTER
                        MOV#    3,3 SNR ; IF ZERO,
                        JMP     NOPE    ; LIST IS EMPTY

                        LDA 2   2       ; RESTORE AC2
                        LDA 1   TKT,2   ; TICKET NUMBER

                LOOP:   LDA 0   REF,3   ; TICKET NUMBER
                        SEQ     0,1     ; IF EQUAL,
                        JMP     NOT     ; VERIFY BLOCK

                        LDA 0   BLK,3   ; BLOCK NUMBER
                        MOV#    0,0 SNR ; IF ZERO,
                        JMP     GOT     ; EUREKA!

                NOT:    LDA 3   LINK,3  ; NEXT TICKET
                        LDA 0   LIST    ; STARTING POINT
                        SEQ     0,3     ; IF NOT EQUAL,
                        JMP     LOOP    ; KEEP LOOKING

                        ; ALLOCATE BLOCK SPACE

                NOPE:   SUBZL   0,0     ; TYPE = 1
                        STA 0   BLK1    ; SET TYPE
                        LDA 2   .IOC1   ; IOCB ADDRESS
                        .IO             ; ALLOCATE
                        JMP     ALL-1   ; TRY AGAIN

                        LDA 2   2       ; RESTORE AC2
                        LDA 1   BLK1    ; BLOCK ADDRESS
```



```
                        ; WAIT FOR THE DISK

00202'102520            SUBZL   0,0     ; DISK = 1
00203'004050            ENQ             ; GRAB IT

00204'044740            STA 1   BLK2    ; BLOCK ADDRESS
00205'020721            LDA 0   .FWD    ; BUFFER OFFSET
00206'107000            ADD     0,1     ; BUFFER ADDRESS
00207'044736            STA 1   SAVE    ; SAVE FOR LATER

                        ; SEARCH FOR FILE

00210'024722            LDA 1   .FN
00211'133000            ADD     1,2     ; FILE NAME
00212'034041            LDA 3   DOOS    ; SYSTEM TABLE
00213'007402            JSR     @D.SRC,3; CALL SEARCH
00214'000440            JMP     NONE    ; NO FILE

                        ; READ IN A SEGMENT

00215'021010            LDA 0   F.ID,2  ; FILE IDENTIFIER
00216'041454            STA 0   D.SP3,3 ; SAVE FOR PRIME
00217'021013            LDA 0   F.FST,2 ; HIGH-ORDER ADDRESS
00220'025015            LDA 1   F.BFP,2 ; LOW-ORDER ADDRESS
00221'030707            LDA 2   .377    ; HALF-WORD MASK
00222'101220            MOVZR   0,0     ; SHIFT HIGH-ORDER, THEN
00223'147000            ANDL    2,1     ; LOW-ORDER, WITH CARRY
00224'030721            LDA 2   SAVE    ; BUFFER ADDRESS
00225'005753            JSR     @D.PRM,3 ; PRIME THE BUFFER

                        ; RELEASE THE DISK

00226'102520            SUBZL   0,0     ; DISK = 1
00227'004052            DEQ             ; GIVE IT UP
```

```
                ; ADD TO BLOCK LIST

                        LDA 3   BLK2    ; BLOCK ADDRESS
                        LDA 2   LIST    ; LIST POINTER
                        LDA 1   LINK,2  ; FOLLOW LINK
                        MOV#    2,0 SNR ; IF NO LIST,
                        MOV     3,2 SKP ; MAKE ONE

                        STA 1   LINK,3  ; LINK TO REAR
                        STA 3   LINK,2  ; AND TO FRONT
                        STA 3   LIST    ; UPDATE LIST
                        LDA 2   2       ; RESTORE AC2

                        SUB     0,0
                        STA 0   MOD,3   ; CLEAR UPDATE
                        STA 0   USE,3   ; AND USE-COUNT
                        LDA 1   TKT,2   ; TICKET NUMBER
                        STA 1   REF,3   ; FOR REFERENCE
                        STA 0   BLK,3   ; SET BLOCK ZERO

                ; TICKET NOW PRESENT

                GOT:    ISZ     USE,3   ; BUMP USE-COUNT
                        STA 3   DATA,2  ; BLOCK ADDRESS
                        LDA 0   .FORM   ; TICKET FORMAT
                        STA 0   FORM,2  ; INTO THE TIT
                        JMP     @RA,2   ; RETURN

                ; FILE NOT FOUND

                NONE:   LDA 2   .IOC2   ; IOCB ADDRESS
                        .IO             ; DEALLOCATE
                        ABORT           ; WILL NEVER OCCUR
                        LDA 2   2       ; RESTORE AC2
                        SUBZL   0,0     ; DISK = 1
                        DEQ             ; RELEASE IT

                        SUB     0,0
                        STA 0   DATA,2  ; NO DATA BLOCK
                        LDA 0   OLD,2   ; PREVIOUS TICKET
                        STA 0   TKT,2   ; BECOMES CURRENT
                        JMP     ERROR   ; COMPLAIN

                .END
```


SYMBOL	VALUE	DEF'N	REFERENCES					
ALL	000154'	6:04	3:41	6:30				
ASK	$X	1:08	2:12					
BLK	$X	5:34	6:15	8:19				
BLK1	000136'	5:12	6:27	6:33				
BLK2	000144'	5:19	7:06	8:03				
CON	$X	1:25	4:24					
D1	000047	1:34	2:29	3:38	4:18			
D2	000046	1:33	2:23	3:34	4:12			
D3	000045	1:32	2:25	3:32	4:14			
D4	000044	1:31	2:18	3:23	4:07			
D5	000043	1:30	2:20	3:26	4:09			
DATA	$X	1:23	3:13	8:24	8:39			
END	$X	1:28	1:54					
ERR	000146'	5:26	4:30					
ERROR	000121'	4:30	3:07	8:42				
F1	000100	1:44	3:30					
F2	000101	1:45	3:36					
F3	000102	1:46	3:40					
FAULT.	000040'	3:03	1:48	2:32				
FIND.	000000'	1:51	1:48					
FI	000100	1:41	1:44	1:45	1:46	5:07		
FORM	$X	1:24	8:26					
GET	000056'	3:23	3:16					
GOT	000247'	8:23	6:17					
LAST	000062	1:39	1:56					
LIB	$X	1:22	6:19	8:05	8:09	8:10		
LIST	$X	1:14	3:18	6:04	6:20	8:04	8:11	
LOOP	000161'	6:11	6:22					
MOD	$X	5:31	8:15					
MSG	000147'	5:27	2:08					
NONE	000254'	8:31	7:17					
OPE	000173'	6:26	6:36					
OUT	000167'	6:19	6:13					
OLD	$X	1:27	3:05	3:11	8:40			
RA	000051	1:37	3:03	8:27				
REF	000050	1:36	1:51	2:13	2:34	4:04	4:25	4:32
REF	$X	5:33	6:11	8:18				
SAVE	000145'	5:21	7:09	7:28				
STAT	000175'	4:43	1:53					
TAD	$X	1:17	2:04					
TKT	$X	1:26	3:09	3:23	6:09	8:17	8:41	
TN	000110	1:42	2:17	2:22	2:27	4:06	4:11	4:16 4:21
			5:06					
TYPE	$X	1:07	2:07					
USE	$X	5:32	3:15	8:16	8:23			
ZAP	$X	1:21	2:05	4:33				
.130	000127'	5:03	4:31					
.377	000130'	5:04	7:25					
.5	000125'	5:01	2:09					
.CVA	$X	1:09	3:24					
.EX	$X	1:10	2:31	4:20				
.IO	$X	1:12	2:33					
.FN	000132'	5:07	7:13					
.FORM	$X	1:13	8:25					
.END	000126'	5:02	7:07					

VALUE	DEFN	REFERENCES
2133'	5:09	6:28
1417	5:16	8:31
$X	1:15	1:52
2131'	5:06	2:14
$X	1:16	2:35
$X	1:11	4:26

FLAGGED LINES: NONE

TICKET PAGING (PAGE.)

The ticket paging routine will locate and display additional pages of the current ticket. If the required data block is not in core, it is read in from disk and added to the block list. If it is already present, its use count is incremented. Note that the back side of a ticket requires a different format block than the front side. Output to the display makes use of the format display routine .DUMP.


```
                        .HEAD   A I D S    T I C K E T    P A G I N G
                        .TITLE  PAGE.
                        ;       AIDS TABLE ASSIGNMENTS
                        .EXTD   .DUMP   ; FORMATTED BLOCK OUTPUT
                        .EXTD   .BACK   ; FORMAT BLOCK ADDRESS
                        .EXTD   LIST    ; BLOCK LIST POINTER
                        .EXTD   .FIND   ; LOCATE TICKET BLOCK
                        .EXTD   .TOP    ; INITIATE EDIT MODE
                        .EXTD   .KILL   ; DELETE CODE BLOCK
                        ;       TIT WORD ASSIGNMENTS
                        .EXTD   ZAP     ; DISPLAY OUTPUT ENTRY
                        .EXTD   LINK    ; GENERAL LINK WORD
                        .EXTD   DATA    ; DATA BLOCK ADDRESS
                        .EXTD   FORM    ; FORMAT BLOCK ADDRESS
                        .EXTD   TKT     ; TICKET NUMBER
          000050        RC=     R0      ; RETURN ADDRESS
          000051        RA=     R1      ; ALTERNATE RETURN
          000060        FID=    S0      ; FILE IDENTIFIER
          000061        FBN=    S1      ; FILE BLOCK NUMBER
          000062        FLA=    S2      ; LOW-ORDER ADDRESS
          000063        FHA=    S3      ; HIGH-ORDER ADDRESS
                        .ENT    PAGE.,PALT.
                        .NREL
000000'055050  PAGE.:  STA 3   RC,2    ; SAVE RETURN
000001'006006$         JSR     @.KILL  ; DELETE CODE
000002'004405          JSR     PALT.   ; FIND NEXT BLOCK
000003'006004$         JSR     @.FIND  ; NO NEXT BLOCK
000004'006001$         JSR     @.DUMP  ; AND DISPLAY IT
000005'035051          LDA 3   RC,2    ; RETURN ADDRESS
000006'002005$         JMP     @.TOP   ; EXIT THRU TOP
```

```
                        ; ALTERNATE ENTRY

                PALT.:  STA 3   RA,2    ; SAVE RETURN
                        LDA 3   DATA,2  ; CURRENT BLOCK
                        MOV#    3,3 SNR ; IF ZERO,
                        JMP     ERROR   ; COMPLAIN
                        LDA 1   TKT,2   ; TICKET NUMBER

                        LDA Ø   T.FOR,3 ; FORE-POINTER
                        COMZR#  Ø,Ø SNR ; IF ALL-ONES,
                        JMP     @RA.2   ; NO NEXT BLOCK
                        STA Ø   FHA.2   ; SAVE FOR PRIME
                        LDA Ø   T.BFP,3 ; BACK/FORE PTRS
                        STA Ø   FLA.2   ; SAVE THIS, TOO
                        LDA Ø   T.ID,3  ; FILE IDENTIFIER
                        STA Ø   FID.2   ; PRIME NEEDS THIS

                        LDA Ø   BLK,3   ; BLOCK NUMBER
                        INC     Ø,Ø     ; BUMP TO NEXT
                        STA Ø   FBN,2   ; SAVE THIS, TOO
                        DSZ     USE,3   ; REDUCE COUNT
                        JMP     ALL     ; CONTINUE

                        STA 3   LIST    ; NEXT IN LINE
                        IDLE            ; IN CASE OF SKIP

                        ; SEARCH BLOCK LIST

                ALL:    LDA 3   LIST    ; LIST POINTER
                        MOV#    3,3 SNR ; IF ZERO,
                        JMP     NOPE    ; LIST IS EMPTY
                        LDA 2   2       ; RESTORE AC2

                LOOP:   LDA Ø   REF,3   ; TICKET NUMBER
                        LDA 1   TKT,2   ; DESIRED TICKET
                        SEQ     Ø,1     ; IF EQUAL,
                        JMP     NOT     ; VERIFY BLOCK

                        LDA Ø   BLK,3   ; BLOCK NUMBER
                        LDA 1   FBN,2   ; DESIRED BLOCK
                        SNE     Ø,1     ; IF EQUAL,
                        JMP     GOT     ; EUREKA!

                NOT:    LDA 3   LINK,3  ; NEXT BLOCK
                        LDA Ø   LIST    ; STARTING POINT
                        SEQ     Ø,3     ; IF NOT EQUAL,
                        JMP     LOOP    ; KEEP LOOKING

                        ; ALLOCATE BLOCK SPACE

                NOPE:   SUBZL   Ø,Ø     ; TYPE = 1
                        STA Ø   BLK1    ; SET TYPE
                        LDA 2   .IOC1   ; IOCB ADDRESS
                        .IO             ; ALLOCATE
                        JMP     ALL-1   ; TRY AGAIN

                        LDA 2   2       ; RESTORE AC2
                        LDA 1   BLK1    ; BLOCK ADDRESS
```



```
                        ; WAIT FOR THE DISK

00062 102520            SUBZL   Ø,Ø     ; DISK = 1
00063 004050            ENQ             ; GRAB IT

00064 044464            STA 1   BLK2    ; BLOCK ADDRESS
00065 020453            LDA Ø   .END    ; BUFFER OFFSET
00066 107000            ADD     Ø,1     ; BUFFER ADDRESS
00067 044462            STA 1   SAVE    ; SAVE FOR LATER

                        ; READ NEXT SEGMENT

00070 034041            LDA 3   DDOS    ; SYSTEM ADDRESS
00071 021060            LDA Ø   FID.2   ; FILE IDENTIFIER
00072 041454            STA Ø   D.SP3,3 ; SAVE FOR PRIME
00073 021063            LDA Ø   FHA.2   ; HIGH-ORDER ADDRESS
00074 025062            LDA 1   FLA.2   ; LOW-ORDER ADDRESS
00075 030444            LDA 2   .377    ; HALF-WORD MASK
00076 101220            MOVZR   Ø,Ø     ; SHIFT HIGH-ORDER, THEN
00077 147000            ANDL    2,1     ; LOW-ORDER, WITH CARRY
00100 030451            LDA 2   SAVE    ; BUFFER ADDRESS
00101 153240            ADDOR   2,2     ; SET HIGH-ORDER BIT
00102 005753            JSR     D.PRM,3 ; PRIME THE BUFFER

                        ; RELEASE THE DISK

00103 102520            SUBZL   Ø,Ø     ; DISK = 1
00104 004052            DEQ             ; GIVE IT UP

                        ; ADD TO BLOCK LIST

00105 034443            LDA 3   BLK2    ; BLOCK ADDRESS
00106 030003            LDA 2   LIST    ; LIST POINTER
00107 025010            LDA 1   LINK,2  ; FOLLOW LINK
00110 141015            MOV#    2,Ø SNR ; IF NO LIST,
00111 171001            MOV     3,2 SKP ; MAKE ONE

00112 045410            STA 1   LINK,3  ; LINK TO REAR
00113 055010            STA 3   LINK,2  ; AND TO FRONT
00114 054003            STA 3   LIST    ; UPDATE LIST
00115 030002            LDA 2   2       ; RESTORE AC2

00116 102400            SUB     Ø,Ø
00117 041414            STA Ø   MOD,3   ; CLEAR UPDATE
00120 041415            STA Ø   USE,3   ; AND USE-COUNT
00121 025013            LDA 1   TKT,2   ; TICKET NUMBER
00122 045416            STA 1   REF,3   ; FOR REFERENCE
00123 025061            LDA 1   FBN,2   ; BLOCK NUMBER
00124 045417            STA 1   BLK,3   ; AS WELL

                        ; BLOCK NOW PRESENT

00125 011415            GOT:    ISZ     USE,3   ; BUMP USE-COUNT
00126 055011            STA 3   DATA,2  ; BLOCK ADDRESS
00127 020002            LDA Ø   .BACK   ; FORMAT ADDRESS
00130 041012            STA Ø   FORM,2  ; INTO THE TIT
00131 035051            LDA 3   RA,2    ; RETURN ADDRESS
00132 001401            JMP     1,3     ; SKIP RETURN
```

```
                ; INDICATE ERROR

 0133/024404    ERROR:  LDA 1   .7      ; BELL CODE
 0134/121401            INC     1,0     ; SET FLAGS
 0135/035050            LDA 3   RC,2    ; RETURN ADDRESS
 0136/003377            JMP     @ZAP,2  ; EXIT THRU ERROR

 0137/000007    .7:     7
 0140/000011    .FWD:   T.FW
 0141/000377    .377:   377

 0142/000143    .IOC1:  .+1     ; ALLOCATE IOCB
 0143/000000            0
 0144/000000            0
 0145/000000    BLK1:   0
 0146/000000            0
 0147/000042            T.FW+B.LEN+1

 0150/000000    BLK2:   0
 0151/000000    SAVE:   0

                ; TICKET DATA BLOCK ASSIGNMENTS

                .EXTD   MOD     ; DATA BLOCK UPDATE FLAG
                .EXTD   USE     ; DATA BLOCK USAGE COUNT
                .EXTD   REF     ; TICKET NUMBER IN DATA
                .EXTD   BLK     ; BLOCK NUMBER IN TICKET

                ; DATA BLOCK/BUFFER PREFIX OFFSETS

 000022         T.ID=   T.FW+B.ID       ; FILE IDENTIFIER
 000017         T.FOR=  T.FW+B.FOR      ; FORE POINTER
 000021         T.BEP=  T.FW+B.BEP      ; BACK/FORE PTR

                .END
```


SYMBOL	VALUE	DEF'N	REFERENCES				
ALL	000033'	2:29	2:22	2:55			
BLK	$X	4:27	2:18	2:39	3:49		
BLK1	000145'	4:15	2:52	2:58			
BLK2	000150'	4:19	3:06	3:32			
DATA	$X	1:18	2:04	3:54			
ERROR	000133'	4:03	2:05				
FBN	000061	1:26	2:20	2:40	3:48		
FBA	000063	1:28	2:12	3:16			
FID	000060	1:25	2:16	3:14			
FLA	000062	1:27	2:14	3:17			
FOR	$X	1:19	3:56				
GOT	000125'	3:53	2:42				
LINK	$X	1:17	2:44	3:34	3:38	3:39	
LIST	$X	1:09	2:24	2:29	2:45	3:33	3:40
LOOP	000037'	2:34	2:47				
MOD	$X	4:24	3:44				
MORE	000053'	2:51	2:31				
NOT	000047'	2:44	2:37				
PAGE.	000001'	1:33	1:30				
PALT.	000007'	2:03	1:33	1:35			
RA	000051	1:23	2:03	2:11	3:57		
RC	000050	1:22	1:33	1:38	4:05		
REF	$X	4:26	2:34	3:47			
SAVE	000151'	4:20	3:09	3:21			
TKT	$X	1:20	2:07	2:35	3:46		
T.	000021	4:33	2:13				
T.FOR	000017	4:32	2:09				
T.ID	000022	4:31	2:15				
USE	$X	4:25	2:21	3:45	3:53		
ZAP	$X	1:16	4:06				
.377	000141'	4:10	3:18				
.7	000137'	4:08	4:03				
.BACK	$X	1:08	3:55				
.DUMP	$X	1:07	1:37				
.FIND	$X	1:10	1:36				
.FWD	000140'	4:09	3:07				
.IOC1	000142'	4:12	2:53				
.KILL	$X	1:12	1:34				
.POP	$X	1:11	1:39				

FLAGGED LINES: NONE

APPEND UTILITY (ADD.)

The append utility is used to add additional pages to a current ticket. It allocates a block of disk from the free chain and adds a new block to the block list. The block is filled with blanks and the new blank page is displayed with the .DUMP routine. Control is transferred to the format edit routine to allow the operator to fill in the blank field.

```
PAGE  1        A I D S    A P P E N D    U T I L I T Y
02/20/74
 1                      .HEAD   A I D S    A P P E N D    U T I L I T Y
 2
 3                      .TITLE  ADD.
 4
 5                      ;       AIDS TABLE ASSIGNMENTS
 6
 7          000110      .DUMP=  110     ; FORMATTED BLOCK OUTPUT
 8          000113      .BACK=  113     ; ADD-ON TICKET FORMAT
 9          000116      .TOP=   116     ; ENTER FORMAT EDIT MODE
10          000133      .PAGE=  133     ; LOCATE NEXT TICKET PAGE
11          000144      LIST=   144     ; DATA BLOCK LIST POINTER
12
13                      ;       TIT WORD ASSIGNMENTS
14
15          000002      LINK=   2       ; GENERAL LINK WORD
16          000006      ZAP=    6       ; DISPLAY OUTPUT ENTRY
17          000014      FORM=   14      ; FORMAT BLOCK ADDRESS
18          000015      DATA=   15      ; DATA BLOCK ADDRESS
19          000024      TKT=    24      ; TICKET NUMBER
20
21          000053      RA=     R3      ; RETURN ADDRESS
22
23          000064      FID=    S4      ; FILE IDENTIFIER
24          000065      FBN=    S5      ; FILE BLOCK NUMBER
25          000066      NEXT=   S6      ; NEW BLOCK ADDRESS
26          000067      SAVE=   S7      ; POINTER SAVE WORD
27
28                      .NREL
29
30                      ; VERIFY TICKET
31
32  000000'055053   ADD.:   STA 3   RA,2     ; SAVE RETURN
33  000001'021015           LDA 0   DATA,2  ; CURRENT BLOCK
34  000002'101014           MOV#    0,0 SZR ; IF NOT ZERO,
35  000003'000405           JMP     GET     ; CONTINUE
36
37  000004'024557           LDA 1   ERR     ; TYPE "?"
38  000005'020557           LDA 0   .130
39  000006'003006           JMP     @ZAP,2  ; EXIT THRU ZAP
40
41                      ; ALLOCATE SPACE
42
43  000007'006040           IDLE
44  000010'102520   GET:    SUBZL   0,0     ; TYPE = 1
45  000011'040563           STA 0   BLOCK   ; SET TYPE
46  000012'030557           LDA 2   .IOCB   ; IOCB ADDRESS
47  000013'006044           .IO             ; ALLOCATE
48  000014'000773           JMP     GET-1   ; TRY AGAIN
49
50  000015'030002           LDA 2   2       ; RESTORE AC2
51  000016'024556           LDA 1   BLOCK   ; BLOCK ADDRESS
52  000017'045066           STA 1   NEXT,2  ; SAVE FOR LATER
```

```
                        ; LOCATE LAST PAGE

                NOT:    JSR     @.PAGE  ; FIND NEXT PAGE
                        JMP     GOT     ; FOUND LASE ONE
                        JMP     NOT     ; KEEP LOOKING

                        ; WAIT FOR THE DISK

                GOT:    SUBZL   0,0     ; DISK = 1
                        ENQ             ; GRAB IT

                        LDA 1   NEXT,2  ; BLOCK ADDRESS
                        LDA 2   .FNB    ; BUFFER OFFSET
                        ADD     1,2     ; BUFFER ADDRESS

                        ; READ FREE SECTOR

                        LDA 3   DDOS    ; SYSTEM ADDRESS
                        SUB     0,0     ; FREE CHAIN = 0
                        STA 0   D.SP3,3 ; SET FOR PRIME
                        LDA 0   D.FR0,3 ; GRAB BOTH PARTS
                        LDA 1   D.FR1,3 ; OF DISK ADDRESS
                        ADDOR   2,2     ; SET OPTION BIT
                        JSR     D.PRM,3 ; PRIME THE BUFFER
                        LDA 2   2       ; RESTORE AC2

                        ; RELEASE THE DISK

                        SUBZL   0,0     ; DISK = 1
                        DEQ             ; GIVE IT UP

                        ; VERIFY LAST PAGE

                        LDA 3   DATA,2  ; OLD BLOCK ADDRESS
                        LDA 0   T.FOR,3 ; OLD FORE POINTER
                        COMZR#  0,0 SZR ; IF NOT ALL-ONES,
                        JMP     NOT     ; START OVER AGAIN

                        ; SAVE FILE INFO

                        LDA 0   T.BFP,3 ; BACK/FORE POINTER
                        LDA 1   .177400 ; HIGH-BYTE MASK
                        AND     1,0     ; EXTRACT HIGH PART
                        STA 0   SAVE,2  ; SAVE BACK POINTER

                        LDA 0   T.ID,3  ; FILE IDENTIFIER
                        STA 0   FID,2   ; SAVE FOR LATER
                        LDA 0   BLK,3   ; THIS BLOCK NUMBER
                        INC     0,0     ; INCREMENT IT
                        STA 0   FBN,2   ; NEXT BLOCK NUMBER
```



```
                        ; UPDATE FREE CHAIN

00057'035066                    LDA 3   NEXT,2  ; FREE BLOCK ADDRESS
00060'021417                    LDA 0   T.FOR,3 ; NEW FREE POINTER
00061'025421                    LDA 1   T.BFP,3 ; HIGH AND LOW

00062'034503                    LDA 3   .377    ; LOW-BYTE MASK
00063'101220                    MOVZR   0,0     ; SHIFT HIGH PART, THEN
00064'167500                    ANDL    3,1     ; LOW PART, WITH CARRY

00065'034041                    LDA 3   DDOS    ; SYSTEM ADDRESS
00066'041421                    STA 0   D.FR0,3 ; FREE CHAIN POINTER
00067'045422                    STA 1   D.FR1,3 ; HIGH AND LOW PARTS
00070'035066                    LDA 3   NEXT,2  ; NEW BLOCK ADDRESS

                        ; UPDATE OLD HEADER

00071'021414                    LDA 0   T.DA0,3 ; DISK ADDRESS
00072'025415                    LDA 1   T.DA1,3 ; HIGH AND LOW
00073'125220                    MOVZR   1,1     ; SHIFT LOW PART, THEN
00074'101100                    MOVL    0,0     ; HIGH PART, WITH CARRY
00075'035067                    LDA 3   SAVE,2  ; GET BACK POINTER
00076'167000                    ADD     3,1     ; ADD TO NEW FORE

00077'035015                    LDA 3   DATA,2  ; OLD BLOCK ADDRESS
00100'041417                    STA 0   T.FOR,3 ; FORE POINTER
00101'045421                    STA 1   T.BFP,3 ; HIGH AND LOW
00102'011407                    ISZ     MOD,3   ; SET UPDATE FLAG
00103'015406                    DSZ     USE,3   ; RELEASE OLD BLOCK
00104'000401                    NOP             ; (IN CASE OF SKIP)

                        ; CREATE NEW HEADER

00105'021414                    LDA 0   T.DA0,3 ; DISK ADDRESS
00106'025415                    LDA 1   T.DA1,3 ; HIGH AND LOW
00107'125220                    MOVZR   1,1     ; SHIFT LOW PART, THEN
00110'101100                    MOVL    0,0     ; HIGH PART, WITH CARRY
00111'034451                    LDA 3   FULL    ; = 512. * 2
00112'167300                    ADDS    3,1     ; ADD TO BACK PTR

00113'035066                    LDA 3   NEXT,2  ; NEW BLOCK ADDRESS
00114'041420                    STA 0   T.BCK,3 ; BACK POINTER
00115'045421                    STA 1   T.BFP,3 ; HIGH AND LOW
00116'102120                    ADCZL   0,0     ; = 177776
00117'041417                    STA 0   T.FOR,3 ; INDICATES EOF
00120'021064                    LDA 0   FID,2   ; FILE IDENTIFIER
00121'041422                    STA 0   T.ID,3  ; HEADER COMPLETE
```

```
                ; ADD TO BLOCK LIST

                        LDA 2   LIST    ; LIST POINTER
                        LDA 1   LINK,2  ; FOLLOW LINK
                        MOV#    2,0 SNR ; IF NO LIST,
                        MOV     3,2 SKP ; MAKE ONE

                        STA 1   LINK,3  ; LINK TO REAR
                        STA 3   LINK,2  ; AND TO FRONT
                        STA 3   LIST    ; UPDATE LIST
                        LDA 2   2       ; RESTORE AC2

                        SUBZL   1,1     ; USERS = 1
                        STA 1   MOD,3   ; SET UPDATE FLAG
                        STA 1   USE,3   ; AND USAGE COUNT
                        LDA 1   TKT,2   ; TICKET NUMBER
                        STA 1   REF,3   ; FOR REFERENCE
                        LDA 1   BLK,2   ; BLOCK NUMBER
                        STA 1   BLK,3   ; AS WELL

                ; BLOCK NOW PRESENT

                        STA 3   DATA,2  ; BLOCK ADDRESS
                        LDA 0   .BACK   ; FORMAT ADDRESS
                        STA 0   FORM,2  ; INTO THE TIT

                ; CLEAR TO ALL BLANKS

                        LDA 0   .FWD    ; DATA OFFSET
                        ADD     0,3     ; MOVE TO START
                        LDA 0   COUNT   ; BLANK COUNTER
                        LDA 1   SPACE   ; ASCII SPACE-CODE
                        STA 1   0,3     ; CLEAR A WORD
                        INC     3,3     ; BUMP ADDRESS
                        INC     0,0 SZR ; AND COUNTER
                        JMP     .-3     ; IF MORE, REPEAT

                ; DISPLAY FOR EDIT

                        JSR     @.DUMP  ; DISPLAY BLOCK
                        LDA 3   .TOP    ; TOP ENTRY ADDRESS
                        ISZ     RA,2    ; BUMP RETURN ADDRESS
                        JMP     @RA,2   ; ISSUE SKIP RETURN
```



```
00160'020040    SPACE:  " *401
00161'177740    COUNT:  B.FWD-B.LEN
00162'175000    FULL:   5**.*204
00163'003477    ERR:    7*4*0+"?

00164'000130    .130:   130
00165'000377    .377:   377
00166'177400    .1774:  377*400

00167'000010    .FWB:   T.FW
00170'000025    .FWD:   T.FW+B.FWD

00171'000172'   .IOCB:  .+1     ; ALLOCATE IOCB
00172'000000            0
00173'000000            0
00174'000000    BLOCK:  0
00175'000000            0
00176'000420            T.FW+B.LEN+1

                ; TICKET DATA BLOCK ASSIGNMENTS

        000004  REF=    4       ; TICKET NUMBER IN BLOCK
        000005  BLK=    5       ; BLOCK NUMBER IN TICKET
        000006  USE=    6       ; DATA BLOCK USAGE COUNT
        000007  MOD=    7       ; DATA BLOCK UPDATE FLAG

                ; DATA BLOCK/BUFFER PREFIX OFFSETS

        000014  T.DA0=  T.FW+B.DA0      ; DISK ADDRESS
        000015  T.DA1=  T.FW+B.DA1      ; HIGH AND LOW
        000017  T.FOR=  T.FW+B.FOR      ; FORE POINTER
        000020  T.BCK=  T.FW+B.BCK      ; BACK POINTER
        000021  T.BFP=  T.FW+B.BFP      ; BACK/FORE PTR
        000022  T.ID=   T.FW+B.ID       ; FILE IDENTIFIER

        000000' .END    ADD.
```

X.	VALUE	DEF'N	REFERENCES			
		1:32	5:36			
		5:23	2:48	4:19		
		5:16	1:45	1:51		
		5:02	4:31			
		1:18	1:33	2:34	3:25	4:23
		5:04	1:37			
		1:24	2:50	4:18		
		1:23	2:47	3:46		
		1:17	4:26			
		5:03	3:38			
		1:44	1:35	1:48		
		2:09	2:04			
		1:15	4:14	4:08	4:09	
		1:11	4:03	4:10		
		5:05	3:28	4:14		
		1:25	1:52	2:12	3:03	3:14 3:41
		2:13	2:15	2:37		
		1:21	1:32	4:42	4:43	
		5:32	4:17			
		1:26	2:44	3:22		
		5:11	4:32			
		1:17	4:16			
		5:32	3:42			
		5:33	2:41	3:05	3:27	3:43
		5:29	3:13	3:34		
		5:30	3:17	3:35		
		5:31	2:36	3:04	3:26	3:45
		5:34	2:46	3:47		
		5:24	3:29	4:15		
		1:16	1:37			
		5:16	1:33			
		5:10	2:42			
		5:17	3:17			
		1:18	4:24			
		1:17	4:43			
		5:12	2:13			
		5:11	4:29			
		5:13	1:46			
		1:13	2:03			
		1:12	4:41			

FLAGGED LINES: NONE

UNIT FUNCTION (UNIT.)

The unit function asks for a unit number and a ten-code. It creates a file name from the ten-code and initiates the named file with a call to the invoke routine, .CALL. If the ten-code is zero (not blank), the ticket which the input unit is assigned to is made current and displayed by calls to .FIND and .DUMP. The unit number is located in the status table with a call to .US3. The address of the unit status output routine, .USØ, is stored in the end control word for use by the status function .STAT, which will display the status for the indicated unit or, if the current unit is zero, the entire unit status table.


```
                .HEAD   A I D S     U N I T     F U N C T I O N

                .TITLE  UNIT.

000060  HOURS=  60      ; DINOS HOURS WORD
000061  MINS=   61      ; DINOS MINUTES WORD

                ;       AIDS TABLE ASSIGNMENTS

                .EXTD   TYPE    ; TEXT STRING TYPER
                .EXTD   ASK     ; DATA STRING INPUT
                .EXTD   .CVA    ; ASCII CONVERSION
                .EXTD   .CVB    ; BINARY CONVERSION
                .EXTD   .MPY    ; UNSIGNED MULTIPLY
                .EXTD   .USØ    ; UNIT STATUS OUTPUT
                .EXTD   .US3    ; UNIT STATUS LOCATE
                .EXTD   .CALL   ; INVOKE ENTRY POINT
                .EXTD   .DUMP   ; DISPLAY IN FORMAT
                .EXTD   .FIND   ; FIND TICKET BLOCK
                .EXTD   .KILL   ; DELETE CODE BLOCK
                .EXTD   TAD     ; TAG-LINE ADDRESS

                ;       TIT WORD ASSIGNMENTS

                .EXTD   ZAP     ; DISPLAY COMMAND ENTRY
                .EXTD   CCN     ; CONTROL CENTER NAME
                .EXTD   UNIT    ; CURRENT UNIT NUMBER
                .EXTD   END     ; END CONTROL WORD
                .EXTD   TIME    ; CURRENT TIME

000040  CODE=   PØ      ; STATUS CODE

000043  D5=     P3
000044  D4=     P4
000045  D3=     P5      ; DECIMAL DIGITS
000046  D2=     P6
000047  D1=     P7

000050  RC=     RØ      ; RETURN ADDRESS

000062  LAST=   S2      ; LAST UNIT (FOR STATUS)

000110  UN=     110     ; UNIT NUMBER AREA
000114  US=     114     ; UNIT STATUS AREA

                .ENT    UNIT.
                .NREL

000000'055050   UNIT.:  STA 3   RC,2    ; SAVE RETURN
000001'006013$          JSR     @.KILL  ; DELETE CODE
000002'034511           LDA 3   STAT    ; STATUS ENTRY
000003'055021$          STA 3   END,2   ; INTO THE TIT
000004'102400           SUB     0,0
000005'041062           STA 0   LAST,2  ; SET FOR STATUS
```

```
                ; REQUEST PARAMETERS

                        SUB     0,0
                        LDA 1   TAD     ; POSITION
                        JSR     @ZAP,2

                        JSR     @TYPE   ; TYPE "UNIT"
                        MSG1
                        LDA 0   .4
                        LDA 1   .UN     ; UNIT OFFSET
                        ADD     2,1     ; MAKE ADDRESS
                        JSR     @ASK    ; GET UNIT NUMBER
                        JMP     @RC,2   ; IF NONE, EXIT

                        JSR     @TYPE   ; TYPE "CODE"
                        MSG2
                        LDA 0   .4
                        LDA 1   .US     ; CODE OFFSET
                        ADD     2,1     ; MAKE ADDRESS
                        JSR     @ASK    ; GET STATUS CODE
                        JMP     @RC,2   ; IF NONE, EXIT

                ; TIME, UNIT, STATUS

                        LDA 0   HOURS   ; GET THE HOURS
                        LDA 1   .100.
                        JSR     @.MPY   ; TIMES 100.
                        LDA 0   MINS    ; GET THE MINUTES
                        ADD     0,1     ; COMBINE
                        STA 1   TIME,2  ; SAVE TIME

                        LDA 0   UN,2    ; FIRST 2 DIGITS
                        LDA 1   UN+1,2  ; NEXT 2 DIGITS
                        JSR     BIN     ; MAKE BINARY
                        STA 1   UNIT,2  ; UNIT NUMBER

                        LDA 0   US,2    ; FIRST 2 DIGITS
                        LDA 1   US+1,2  ; NEXT 2 DIGITS
                        JSR     BIN     ; MAKE BINARY
                        MOV#    1,1 SNR ; IF ZERO,
                        JMP     FLASH   ; RECALL TICKET

                        LDA 0   .100.
                        LDA 3   .TEN.   ; = 1000.
                        SLE     0,1     ; IF < 100.
                        ADD     3,1     ; ADD 1000.
                        STA 1   CODE,2  ; STATUS CODE
```



```
                ; COMPUTE FILE NAME

                        JSR     @.CVA   ; MAKE ASCII

                        LDA 0   D4,2    ; DIGIT 4
                        MOVS    0,1     ; MOVE TO TOP
                        LDA 0   D3,2    ; DIGIT 3
                        ADD     0,1     ; COMBINE
                        STA 1   FILE1   ; AND SAVE

                        LDA 0   D2,2    ; DIGIT 2
                        MOVS    0,1     ; MOVE TO TOP
                        LDA 0   D1,2    ; DIGIT 1
                        ADD     0,1     ; COMBINE
                        STA 1   FILE2   ; AND SAVE

                ; CALL OVERLAY CODE

                        JSR     @.CALL  ; JUMP INTO INVOKE
                        FILE            ; FILENAME ADDRESS

                ; RECALL ASSIGNED TICKET

FLASH:  LDA 0   CCN,2   ; CONTROL CENTER
                        LDA 1   UNIT,2  ; UNIT NUMBER
                        JSR     @.US3   ; LOCATE IN TABLE
                        JMP     ERROR   ; NOT IN TABLE

                        LDA 1   5,3     ; TICKET NUMBER
                        MOV#    1,1 SZR ; IF ZERO,
                        JMP     @RC,2   ; IGNORE

                        JSR     @.FIND  ; ELSE RECALL
                        JSR     @.DUMP  ; AND DISPLAY
                        JMP     @RC,2

                ; CONVERT TO BINARY

BIN:    STA 1   D1,2    ; DIGIT 1
                        MOVS    1,1
                        STA 1   D2,2    ; AND 2

                        STA 0   D3,2    ; DIGIT 3
                        MOVS    0,0
                        STA 0   D4,2    ; AND 4

                        SUB     0,0     ; DIGIT 5
                        STA 0   D5,2
                        JMP     @.CVB   ; MAKE BINARY
```

```
                        ; DISPLAY STATUS

                STAT:   .+1
                        STA 3   RC,2    ; SAVE RETURN
                        LDA Ø   UN,2
                        LDA 1   UN+1,2  ; UNIT NUMBER
                        JSR     BIN
                        LDA Ø   UN,2
                        MOV#    Ø,Ø SNR ; IF NO INPUT,

                        LDA Ø   CCN,2   ; CONTROL CENTER
                        LDA 3   RC,2
                        JMP     @.USØ   ; STATUS OUTPUT

                        ; TYPE ERROR MESSAGE

                ERROR:  LDA 1   ERR     ; TYPE "?"
                        LDA Ø   .13Ø
                        LDA 3   RC,2    ; RETURN ADDRESS
                        JMP     @ZAP,2  ; EXIT THRU ERROR

                .4:     4
                .13Ø:   13Ø
                .1ØØ.:  1ØØ.
                .TEN.:  1ØØØ.
                .JN:    JN
                .US:    US

                .TXTM   1
                .TXTX   1

                ERR:    7*4ØØ+"?
                MSG1:   .TXT    *UNIT*
                MSG2:   .TXT  *  CODE*

                .TXTN   1

                FILE:   .TXT    *AIDS.UNIT.*
                FILE1:  Ø
                FILE2:  Ø
                        Ø

                        .END
```


SYMBOL	VALUE	DEF'N	REFERENCES						
ASK	$X	1:11	2:12	2:20					
BIN	ØØØ1Ø2'	3:39	2:34	2:39	4:07				
CCN	$X	1:26	3:24	4:11					
CODE	ØØØØ4Ø	1:31	2:47						
D1	ØØØØ47	1:37	3:13	3:39					
D2	ØØØØ46	1:36	3:11	3:41					
D3	ØØØØ45	1:35	3:07	3:43					
D4	ØØØØ44	1:34	3:05	3:45					
D5	ØØØØ43	1:33	3:48						
END	$X	1:28	1:53						
ERR	ØØØ137'	4:32	4:17						
ERROR	ØØØ125'	4:17	3:27						
FILE	ØØØ147'	4:38	3:23						
FILE1	ØØØ154'	4:39	3:09						
FILE2	ØØØ155'	4:40	3:15						
FLASH	ØØØØ7Ø'	3:24	2:41						
HOURS	ØØØØ6Ø	1:05	2:25						
LAST	ØØØØ62	1:41	1:55						
MINS	ØØØØ61	1:06	2:23						
MSG1	ØØØ14Ø'	4:33	2:28						
MSG2	ØØØ143'	4:34	2:16						
RC	ØØØØ5Ø	1:39	1:5Ø	2:13	2:21	3:31	3:35	4:04	4:12
			4:19						
STAT	ØØØ113'	4:03	1:52						
TAD	$X	1:21	2:31						
TI	$X	1:29	2:3[illegible]						
TYPE	$X	1:10	2:07	2:15					
UN	ØØØ11Ø	1:43	2:32	2:33	4:05	4:06	4:38	4:26	
UNIT	$X	1:27	2:35	3:25					
UNIT.	ØØØØØØ'	1:5Ø	1:45						
US	ØØØ114	1:44	2:37	2:38	4:27				
ZAP	$X	1:25	2:05	4:20					
.1ØØ.	ØØØ133'	4:24	2:26	2:43					
.13Ø	ØØØ132'	4:23	4:18						
.4	ØØØ131'	4:22	2:09	2:17					
.CALL	$X	1:17	3:19						
.CVA	$X	1:12	3:03						
.CVB	$X	1:13	3:49						
.DUMP	$X	1:18	3:34						
.FIND	$X	1:19	3:33						
.KILL	$X	1:20	1:51						
.MPY	$X	1:14	2:27						
.TEN.	ØØØ134'	4:25	2:44						
.JN	ØØØ135'	4:26	2:14						
.US	ØØØ136'	4:27	2:18						
.USØ	$X	1:15	4:13						
.US3	$X	1:16	3:26						

FLAGGED LINES: NONE

PROGRAMMING MANUAL
(Part II)

Computer Aided Dispatch System
for the
Police Departments of Oak Park,
River Forest, and Forest Park, Illinois

by

William Behr and Warner Brigham

Center for Advanced Computation
University of Illinois at Urbana-Champaign
Urbana, Illinois 61801

March 1, 1974

This work was supported by a grant from the Illinois Law Enforcement Commission to the Village of Oak Park through an Agreement for Cooperative Investigation between Oak Park and the Board of Trustees of the University of Illinois.

© 1973 by the Board of Trustees of the University of Illinois

FOREWORD

The following report gives a description and the detailed line code for each of the software modules required to make the Automated Interactive Dispatch System operable. This Automated Interactive Dispatch Software System is composed of many software modules which interact with each other mainly through the system vector tables and terminal information tables. The hardware required for the software module to operate is on a Data General Nova 820 computer, a Century #114 disc pack system, and a Decision disc controller. They operate under DINOS (Decision Incorporated Nova Operating System) and DDOS (Decision Disc Operating System). It is essential that these operating systems are functional on the system before the Automated Interactive Dispatch Software System will operate.

The reader of these reports should be familiar with the System Manual of the Computer Aided Dispatch System before looking at this Programming Manual. He should also be familiar with the Data General Nova Computers, their assembly language coding procedure, and somewhat familiar with the two operating systems from Decision, Inc. Each module has a short description preceding the code and the detailed code has comments on each line of code describing exactly each line of code.

TEN-41 UTILITY (TEN41)

The TEN-41 utility is used to put a unit on duty. This utility is initiated by a call to the invoke function from the unit function. The utility will complain if the unit is already on duty or if the control center name is invalid. The utility types the message "NAME" and allows the operator to type in the officer's name or names. The utility creates a new entry in the unit status table and exits through the unit status output routine which displays the new status for this unit.


```
                        .HEAD   A I D S    T E N - 4 1    U T I L I T Y

                        .TITLE  TEN41

                        ;       AIDS TABLE ASSIGNMENTS

        000104  TYPE=    104     ; TEXT STRING TYPER
        000105  ASK=     105     ; DATA STRING INPUT
        000120  .US0=    120     ; UNIT STATUS OUTPUT
        000122  .US1=    122     ; UNIT STATUS INSERT
        000126  .US3=    126     ; UNIT STATUS LOCATE
        000130  .CCN=    130     ; CENTER LOOK-UP
        000137  .LOG=    137     ; MAKE LOG ENTRY
        000134  .CALL=   134     ; CALL OVERLAY

                        ;       TIT WORD ASSIGNMENTS

        000000  ID=      0       ; TERMINAL IDENTIFIER
        000006  ZAP=     6       ; DISPLAY OUTPUT ENTRY
        000020  CCN=     20      ; CONTROL CENTER NAME
        000022  TIME=    22      ; CURRENT TIME OF DAY
        000023  UNIT=    23      ; CURRENT UNIT NUMBER

        000040  CODE=    P0      ; STATUS CODE
        000041  POST=    P1      ; POST NUMBER
        000042  TICK=    P2      ; TICKET NUMBER

        000052  RT=      R2      ; RETURN ADDRESS

        000124  UT=      124     ; STATUS INFO AREA

                        .EXTN   AREA.   ; PATROL AREA LOOKUP

                        .NREL

00000'055052    TEN41:  STA 3   RT,2    ; SAVE RETURN
00001'021020            LDA 0   CCN,2   ; CONTROL CENTER
00002'025023            LDA 1   UNIT,2  ; UNIT NUMBER
00003'125415            MOV#    1,1 SNR ; IF UNIT ZERO,
00004'000442            JMP     ERROR   ; COMPLAIN
00005'006126            JSR     @.US3   ; IF UNIT FOUND
00006'006130            JSR     @.CCN   ; DO BAD CENTER,
00007'000437            JMP     ERROR   ; COMPLAIN

00010'006104            JSR     @TYPE   ; TYPE "AREA"
00011'000057'           MSGI
00012'020443            LDA 0   .30
00013'024441            LDA 1   .UT     ; AREA OFFSET
00014'147000            ADD     2,1     ; MAKE ADDRESS
00015'006105            JSR     @ASK    ; GET PATROL AREA
00016'102401            SUB     0,0 SKP ; NO AREA ENTERED

00017'006430            JSR     @.AREA  ; GET AREA NUMBER
00020'041041            STA 0   POST,2  ; SAVE FOR LATER
```

```
                LDA 0   CCN,2   ; CONTROL CENTER
                LDA 1   UNIT,2  ; UNIT NUMBER
                JSR     @.US1   ; INSERT IN TABLE
                JMP     ERROR   ; NO ROOM IN TABLE

                LDA 0   CODE,2  ; GET THE STATUS
                STA 0   2,3     ; INTO THE TABLE
                LDA 0   TIME,2  ; GET THE TIME
                STA 0   3,3     ; INTO THE TABLE
                LDA 0   POST,2  ; AREA NUMBER
                STA 0   6,3     ; INTO THE TABLE

                LDA 1   .30     ; STRING LENGTH
                MOVOR   1,1     ; MAKE WORD COUNT
JAM:            LDA 0   UT,2    ; OBTAIN A WORD
                STA 0   24,3    ; AND INSERT IT
                INC     3,3     ; BUMP ADDRESS
                INC     2,2     ; THIS ONE TOO
                INC     1,1 SZR ; BUMP COUNTER
                JMP     JAM     ; AND REPEAT

                LDA 2   2       ; RESTORE TIT
                SUB     0,0
                STA 0   POST,2  ; CLEAR POST
                STA 0   TLOG,2  ; AND TICKET
                JSR     @.LOG   ; MAKE LOG ENTRY
                JSR     @.CALL  ; CALL OVERLAY
                FILE            ; LINK TO 10-86

; TYPE ERROR MESSAGE

ERROR:  LDA 1   ERR     ; TYPE "?"
                LDA 0   .130
                LDA 3   RT,2    ; RETURN ADDRESS
                JMP     @ZAP,2  ; EXIT THRU ZAP

.30:    -30
.130:   130
.UT:    UT
.AREA:  AREA.

        .TXT"   1
        .TXTX   1

ERR:    7*400+"?
MSG1:   .TXT    *  AREA*
FILE:   .TXT    *AIDS.UNIT.1086*

        .END    TEN41
```


SYMBOL	VALUE	DEF'N	REFERENCES		
AREA.	'X	1:32	2:40		
ASK	000105	1:08	1:50		
CCN	000020	1:20	1:37	2:01	
CODE	000040	1:24	2:06		
ERR	000056'	2:45	2:32		
ERROR	000046'	2:32	1:40	1:43	2:04
FILE	000063'	2:47	2:28		
ID	000000	1:18			
JAM	000035'	2:15	2:20		
MSG1	000057'	2:46	1:46		
POST	000041	1:25	1:54	2:10	
RT	000052	1:28	1:36	2:34	
TEN41	000000'	1:36	2:49		
TIME	000022	1:21	2:08		
TLOG	000042	1:26			
TYPE	000104	1:07	1:45		
UNIT	000023	1:22	1:38	2:02	
UT	000124	1:30	2:15	2:30	
ZAP	000006	1:19	2:35		
.130	000053'	2:38	2:33		
.30	000052'	2:37	1:47	2:13	
.AREA	000055'	2:40	1:53		
.CALL	000134	1:14	2:27		
.CCN	000130	1:12	1:42		
.LOG	000137	1:13			
.U.	000120	1:09			
.US1	000122	1:10	2:03		
.US3	000126	1:11	1:41		
.UT	000054'	2:39	1:48		

FLAGGED LINES: NONE

TEN-42 UTILITY (TEN42)

The TEN-42 utility is used to take a unit off duty. This utility is initiated by a call to the invoke function from the unit function. The utility will complain if the unit is not on duty or if the unit is assigned to a ticket. The utility displays the status for this unit and then deletes the unit's entry from the unit status table.


```
                        .HEAD   A I D S    T E N - 4 2    U T I L I T Y

                        .TITLE  TEN42

                        ;       AIDS TABLE ASSIGNMENTS

          000120        .US0=   120     ; UNIT STATUS OUTPUT
          000124        .US2=   124     ; UNIT STATUS DELETE
          000126        .US3=   126     ; UNIT STATUS LOCATE
          000137        .LOG=   137     ; MAKE LOG ENTRY

                        ;       TIT WORD ASSIGNMENTS

          000006        ZAP=    6       ; DISPLAY COMMAND ENTRY
          000020        CCN=    20      ; CONTROL CENTER NAME
          000022        TIME=   22      ; CURRENT TIME OF DAY
          000023        UNIT=   23      ; CURRENT UNIT NUMBER

          000040        CODE=   P0      ; STATUS CODE
          000041        POST=   P1      ; POST NUMBER
          000042        TLOG=   P2      ; TICKET NUMBER

          000052        RT=     R2      ; RETURN ADDRESS

          000124        UT=     124     ; STATUS INFO AREA

                        .NREL

00000'055052  TEN42:    STA 3   RT,2    ; SAVE RETURN
00001'021020            LDA 0   CCN,2   ; CONTROL CENTER
00002'025023            LDA 1   UNIT,2  ; UNIT NUMBER
00003'000126            JSR     @.US3   ; LOCATE IN TABLE
00004'000440            JMP     ERROR   ; CAN'T HAPPEN

00005'025405            LDA 1   5,3     ; TICKET NUMBER
00006'125014            MOV#    1,1 SZR ; IF NOT ZERO,
00007'000435            JMP     ERROR   ; COMPLAIN
00010'045042            STA 1   TLOG,2  ; SET FOR LOG
00011'025406            LDA 1   6,3     ; AREA NUMBER
00012'045041            STA 1   POST,2  ; SET FOR LOG
```

```
                LDA 0   CODE,2  ; GET THE STATUS
                STA 0   2,3     ; INTO THE TABLE
                LDA 0   TIME,2  ; GET THE TIME
                STA 0   3,3     ; INTO THE TABLE

                LDA 1   .30     ; STRING LENGTH
                MOVOR   1,1     ; MAKE WORD COUNT
JAM:            LDA 0   10,3    ; OBTAIN A WORD
                STA 0   UT,2    ; AND INSERT IT
                INC     3,3     ; BUMP ADDRESS
                INC     2,2     ; THIS ONE TOO
                INC     1,1 SZR ; BUMP COUNTER
                JMP     JAM     ; AND REPEAT

                LDA 2   2       ; RECOVER TIT
                SUB     0,0
                STA 0   POST,2  ; CLEAR POST
                STA 0   TLOG,2  ; AND TICKET
                JSR     @.LOG   ; MAKE LOG ENTRY
                LDA 0   CCN,2   ; CONTROL CENTER
                LDA 1   UNIT,2  ; UNIT NUMBER
                JSR     @.US0   ; DISPLAY STATUS

                LDA 0   CCN,2   ; CONTROL CENTER
                LDA 1   UNIT,2  ; UNIT NUMBER
                JSR     @.US2   ; DELETE FROM TABLE
                JMP     ERROR   ; CAN'T HAPPEN

                JMP     RT,2    ; RETURN

        ; TYPE ERROR MESSAGE

ERROR:  LDA 1   ERR     ; TYPE "?"
        LDA 0   .130
        LDA 3   RT,2    ; RETURN ADDRESS
        JMP     ZAP,2   ; EXIT THRU ZAP

.30:    -30
.130:   130
ERR:    7*400+"?

        .END    TEN42
```


SYMBOL	VALUE	DEF'N	REFERENCES		
CCN	000020	1:15	1:30	2:20	2:24
CODE	000040	1:19	2:01		
ERR	000052	2:40	2:33		
ERROR	000044	2:33	1:33	1:37	2:27
JAM	000021	2:08	2:13		
POST	000041	1:20	1:40	2:17	
RT	000052	1:23	1:29	2:29	2:35
TEN42	000000	1:29	2:42		
TIME	000022	1:16	2:03		
TLOG	000042	1:21	1:38	2:18	
UNIT	000023	1:17	1:31	2:21	2:25
UT	000124	1:25	2:09		
ZAP	000016	1:14	2:36		
.130	000131	2:39	2:34		
.30	000050	2:38	2:06		
.LOG	000137	1:18	2:19		
.US0	000120	1:17	2:22		
.US2	000124	1:16	2:26		
.US3	000126	1:19	1:32		

FLAGGED LINES: NONE

TEN-44 UTILITY (TEN44)

The TEN-44 utility is used to put a unit on break. This utility is initiated by a call to the invoke function from the unit function. The utility will complain if the unit is not on duty or if the unit is assigned to a ticket. The time and code fields of the entry in the unit status table are updated and the message "LOCATION" is typed. If a new location is typed in by the operator, this information is also updated in the unit status table. The utility exits through the unit status output routine which displays the new status for this unit.


```
                        .HEAD   A I D S   T E N - 4 4   U T I L I T Y
                        .TITLE  TEN44
                        ;       AIDS TABLE ASSIGNMENTS
000104          TYPE=   104     ; TEST STRING TYPER
000105          ASK=    105     ; DATA STRING INPUT
000120          .US0=   120     ; UNIT STATUS OUTPUT
000126          .US3=   126     ; UNIT STATUS LOCATE
000137          .LOG=   137     ; MAKE LOG ENTRY

                        ;       TIT WORD ASSIGNMENTS
000006          ZAP=    6       ; DSIPLAY OUTPUT ENTRY
000020          CCN=    20      ; CONTROL CENTER NAME
000022          TIME=   22      ; CURRENT TIME OF DAY
000023          UNIT=   23      ; CURRENT UNIT NUMBER

000040          CODE=   P0      ; STATUS CODE
000041          POST=   P1      ; POST NUMBER
000042          TLOG=   P2      ; TICKET NUMBER

000052          RT=     R2      ; RETURN ADDRESS

000124          UT=     124     ; STATUS INFO AREA

                        .NREL
000000'021020   TEN44:  LDA 0   CCN,2    ; CONTROL CENTER
000001'025023           LDA 1   UNIT,2   ; UNIT NUMBER
000002'006126           JSR     @.US3    ; LOCATE IN TABLE
000003'000451           JMP     ERROR    ; NOT IN TABLE

000004'025405           LDA 1   5,3      ; TICKET NUMBER
000005'125014           MOV#    1,1 SZR  ; IF NOT ZERO,
000006'000446           JMP     ERROR    ; COMPLAIN
000007'045042           STA 1   TLOG,2   ; SET FOR LOG
000010'025406           LDA 1   6,3      ; AREA NUMBER
000011'045041           STA 1   POST,2   ; SET FOR LOG

000012'006104           JSR     @TYPE    ; TYPE "LOCATION"
000013'000064'          MSG
000014'020444           LDA 0   .30
000015'024445           LDA 1   .UT      ; LOCATION OFFSET
000016'147000           ADD     2,1      ; MAKE ADDRESS
000017'006105           JSR     @ASK     ; GET LOCATION
000020'003052           JMP     @RT,2    ; NO LOCATION
```

```
                        LDA 0   CCN,2   ; CONTROL CENTER
                        LDA 1   UNIT,2  ; UNIT NUMBER
                        JSR     @.US3   ; LOCATE IN TABLE
                        JMP     ERROR   ; NOT IN TABLE

                        LDA 0   CODE,2  ; GET THE STATUS
                        STA 0   2,3     ; INTO THE TABLE
                        LDA 0   TIME,2  ; GET THE TIME
                        STA 0   3,3     ; INTO THE TABLE
                        SUB     0,0
                        STA 1   4,3     ; CLEAR POST

                        LDA 1   .34     ; STRING LENGTH
                        MOVOR   1,1     ; MAKE WORD COUNT
JA%:    LDA 0   UT,2    ; OBTAIN A WORD
                        STA 0   34,3    ; AND INSERT IT
                        INC     3,3     ; BUMP ADDRESS
                        INC     2,2     ; THIS ONE TOO
                        INC     1,1 SZR ; BUMP COUNTER
                        JMP     JA%     ; AND REPEAT

                        LDA 2   2       ; RESTORE TIT
                        SUB     0,0
                        STA 0   POST,2  ; CLEAR POST
                        STA 0   TLOG,2  ; AND TICKET
                        JSR     .LOG    ; MAKE LOG ENTRY
                        LDA 0   CCN,2   ; CONTROL CENTER
                        LDA 1   UNIT,2  ; UNIT NUMBER
                        LDA 3   RT,2    ; RETURN ADDRESS
                        JMP     @.US4   ; EXIT THRU STATUS

        ; TYPE "?" AND MESSAGE

                        LDA 1   "?      ; TYPE "?"
                        LDA 0   .130
                        LDA 3   RT,2    ; RETURN ADDRESS
                        JMP     @ZAP,2  ; EXIT THRU ZAP

.34:    -34
.130:   130
UT:     UT

        .ENTS   1
        .ENTX   1

"?:     7*400+"?
ERROR:  .EXT  *  LOCATION*

        .END    TEN44
```


SYMBOL	VALUE	DEFN	REFERENCES			
ASK	000105	1:08	1:47			
CCN	000020	1:16	1:30	2:01	2:27	
CODE	000040	1:20	2:06			
ERR	000063	2:46	2:34			
ERROR	000054	2:34	1:33	1:37	2:44	
JA%	000035	2:15	2:20			
MSG	000064	2:47	1:43			
POST	000041	1:21	1:44	2:24		
RT	000052	1:24	1:48	2:29	2:36	
TIT	000000	1:30	2:42			
TIME	000022	1:17	2:08			
TLOG	000042	1:22	1:38	2:25		
TYPE	000144	1:07	1:42			
UNIT	000023	1:18	1:31	2:02	2:28	
UT	000124	1:26	2:15	2:41		
ZAP	000006	1:15	2:37			
.130	000061	2:14	2:35			
.34	000060	2:39	1:44	2:13		
.LOG	000137	1:11	2:26			
.US4	000120	1:09	2:31			
.US3	000126	1:10	1:32	2:03		
.UT	000062	2:41	1:45			

FLAGGED LINES: NONE

TEN-86 UTILITY (TEN86)

The TEN-86 utility is used to change the names of officers assigned to a unit without changing its status. The utility will complain if the unit is not on duty. This utility is initiated by a call to the invoke function from the unit function. The message "NAME" is displayed and if the operator types in new officers' names, the proper information is updated in the unit status table entry for this unit. The time is updaied whether or not new name information is given. The utility exits through the unit status output routine which displays the new status for this unit.


```
                .HEAD   A I D S     TEN-86    UTILITY
                .TITLE  TEN86
                ;       AIDS TABLE ASSIGNMENTS
000104          TYPE=   104     ; TEXT STRING TYPER
000105          ASK=    105     ; DATA STRING INPUT
000120          .US0=   120     ; UNIT STATUS OUTPUT
000126          .US3=   126     ; UNIT STATUS LOCATE
000137          .LOG=   137     ; MAKE LOG ENTRY

                ;       TIT WORD ASSIGNMENTS

000006          ZAP=    6       ; DISPLAY OUTPUT ENTRY
000020          CCN=    20      ; CONTROL CENTER NAME
000022          TIME=   22      ; CURRENT TIME OF DAY
000023          UNIT=   23      ; CURRENT UNIT NUMBER

000041          POST=   P1      ; POST NUMBER
000042          TICK=   P2      ; TICKET NUMBER

000052          RT=     R2      ; RETURN ADDRESS

000124          UT=     124     ; STATUS INFO AREA

                .NREL

000000'055052   TEN86:  STA 3   RT,2    ; SAVE RETURN
000001'021020           LDA 0   CCN,2   ; CONTROL CENTER
000002'025023           LDA 1   UNIT,2  ; UNIT NUMBER
000003'006126           JSR     @.US3   ; IF NOT FOUND
000004'000440           JMP     ERROR   ; COMPLAIN

000005'006104           JSR     @TYPE   ; TYPE "NAME"
000006'000534'          MSG2
000007'025441           LDA 0   .30
000010'024442           LDA 1   .UT     ; NAME OFFSET
000011'147001           ADD     2,1     ; MAKE ADDRESS
000012'006105           JSR     @ASK    ; GET OFFICER NAME
000013'000425           JMP     OUT     ; NO CHANGE
```

```
                        LDA 0   CCN,2   ; CONTROL CENTER
                        LDA 1   UNIT,2  ; UNIT NUMBER
                        JSR     @.US3   ; LOCATE IN TABLE
                        JMP     ERROR   ; NOT IN TABLE

                        LDA 1   TIME,2  ; CURRENT TIME
                        STA 1   3,3     ; INTO THE TABLE
                        LDA 1   4,3     ; POST NUMBER
                        STA 1   POST,2  ; SET FOR LOG
                        LDA 1   5,3     ; TICKET NUMBER
                        STA 1   TLOG,2  ; SET FOR LOG

                        LDA 1   .30     ; STRING LENGTH
                        MOVOR   1,1     ; MAKE WORD COUNT
JAM:                    LDA 0   UT,2    ; OBTAIN A WORD
                        STA 0   10,3    ; AND INSERT IT
                        INC     3,3     ; BUMP ADDRESS
                        INC     2,2     ; THIS ONE TOO
                        INC     1,1 SZR ; BUMP COUNTER
                        JMP     JAM     ; AND REPEAT

                        LDA 2   2       ; RESTORE TIT
                        JSR     @.LOG   ; MAKE LOG ENTRY
OUT:                    LDA 0   CCN,2   ; CONTROL CENTER
                        LDA 1   UNIT,2  ; UNIT NUMBER
                        LDA 3   RT,2    ; RETURN ADDRESS
                        JMP     @.US0   ; EXIT THRU STATUS

        ; TYPE ERROR MESSAGE

ERROR:  LDA 1   ERR     ; TYPE "?"
                        LDA 0   .13
                        LDA 3   RT,2    ; RETURN ADDRESS
                        JMP     @ZAP,2  ; EXIT THRU ZAP

.30:    -30
.13:    13
.UT:    UT

        .TXTM   1
        .TXTX   1

        7<4
MSG:    .TXT    *   NAME*

        .END    TEN86
```


SYMBOL	VALUE	DEF'N	REFERENCES			
ASK	000105	1:08	1:40			
CCN	000020	1:16	1:30	2:01	2:24	
ERR	000053	2:43	2:31			
ERROR	000044	2:31	1:33	2:04		
JAM	000030	2:15	2:20			
MSG2	000054	2:44	1:36			
OUT	000040	2:24	1:41			
POST	000041	1:20	2:09			
RT	000052	1:23	1:29	2:26	2:33	
TEN86	000000	1:29	2:46			
TIME	000022	1:17	2:06			
TLOG	000042	1:21	2:11			
TYPE	000104	1:07	1:35			
UNIT	000023	1:18	1:31	2:02	2:25	
UT	000124	1:25	2:15	2:38		
ZAP	000006	1:15	2:34			
.13	000015	2:37	2:32			
.30	000036	2:36	1:37	2:13		
.LOG	000137	1:11	2:23			
.US0	000120	1:09	2:27			
.US3	000126	1:10	1:32	2:03		
.UT	000052	2:38	1:38			

FLAGGED LINES: NONE

TEN-2Ø UTILITY (TEN2Ø)

The TEN-2Ø utility is used to change the location of a unit without changing its status. The utility will complain if the unit is not on duty. The message "LOCATION" is displayed and if the operator types in a new location, the proper information is updated in the unit status table entry for this unit. The time information is updated whether or not a new location is given. The utility exits through the unit status output routine which displays the new status for this unit. This utility is initiated by a call to the invoke function from the unit function.

```
PAGE  1          A I D S    T E N - 2 Ø    U T I L I T Y
02/20/74
                 .HEAD  A I D S    T E N - 2 Ø    U T I L I T Y
                 .TITLE TEN2Ø
                 ;      AIDS TABLE ASSIGNMENTS
000104           TYPE=  104      ; TEST STRING TYPER
000105           ASK=   105      ; DATA STRING INPUT
000120           .US2=  120      ; UNIT STATUS OUTPUT
000126           .US3=  126      ; UNIT STATUS LOCATE
000137           .LOG=  137      ; MAKE LOG ENTRY
                 ;      TIT WORD ASSIGNMENTS
000006           ZAP=   6        ; DSIPLAY OUTPUT ENTRY
000020           CCN=   20       ; CONTROL CENTER NAME
000022           TIME=  22       ; CURRENT TIME OF DAY
000023           UNIT=  23       ; CURRENT UNIT NUMBER
000041           POST=  P1       ; POST NUMBER
000042           TLOC=  P2       ; TICKET NUMBER
000052           RT=    R2       ; RETURN ADDRESS
000124           UT=    124      ; STATUS INFO AREA
                 .NREL
00000'021020     TEN2Ø: LDA Ø   CCN,2    ; CONTROL CENTER
00001'025023            LDA 1   UNIT,2   ; UNIT NUMBER
00002'016126            JSR     @.US3    ; LOCATE IN TABLE
00003'000444            JMP     ERROR    ; NOT IN TABLE
00004'000134            JSP     1TYPE    ; TYPE "LOCATION"
00005'000053'           MSG
00006'020441            LDA Ø   .3Ø
00007'024442            LDA 1   .UT      ; LOCATION OFFSET
00010'147000            ADD     2,1      ; MAKE ADDRESS
00011'016105            JSR     @ASK     ; GET LOCATION
00012'000425            JMP     OUT      ; NO CHANGE
```

```
                LDA 0   CCN,2   ; CONTROL CENTER
                LDA 1   UNIT,2  ; UNIT NUMBER
                JSP     @.US3   ; LOCATE IN TABLE
                JMP     ERROR   ; NOT IN TABLE

                LDA 1   TIME,2  ; CURRENT TIME
                STA 1   3,3     ; INTO THE TABLE
                LDA 1   4,3     ; POST NUMBER
                STA 1   POST,2  ; SET FOR LOG
                LDA 1   5,3     ; TICKET NUMBER
                STA 1   TLOG,2  ; SET FOR LOG

                LDA 1   .30     ; STRING LENGTH
                MOVOR   1,1     ; MAKE WORD COUNT
JAM:            LDA 0   UT,2    ; OBTAIN A WORD
                STA 0   24,3    ; AND INSERT IT
                INC     3,3     ; BUMP ADDRESS
                INC     2,2     ; THIS ONE TOO
                INC     1,1 SZR ; BUMP COUNTER
                JMP     JAM     ; AND REPEAT

                LDA 2   2       ; RESTORE TIT
                JSR     @.LOG   ; MAKE LOG ENTRY
OUT:            LDA 0   CCN,2   ; CONTROL CENTER
                LDA 1   UNIT,2  ; UNIT NUMBER
                LDA 3   RT,2    ; RETURN ADDRESS
                JMP     @.US0   ; EXIT THRU STATUS

        ; TYPE ERROR MESSAGE

ERROR:  LDA 1   EPR     ; TYPE "?"
                LDA 0   .130
                LDA 3   RT,2    ; RETURN ADDRESS
                JMP     0ZAP,2  ; EXIT THRU ZAP

.ASK:   -30
.130:   130
.UT:    UT

        .TXTM   1
        .TXTA   1

EPR:    7*400+"?
RT:     .TXT    * LOCATION*

        .END    TST2
```


SYMBOL	VALUE	DEF'N	REFERENCES		
ASK	000105	1:08	1:39		
CCN	000020	1:16	1:29	2:01	2:24
EPR	000052	2:43	2:31		
ERROR	000043	2:31	1:32	2:04	
JAM	000027	2:15	2:21		
MSG	000053	2:44	1:35		
OUT	000037	2:24	1:40		
POST	000041	1:25	2:09		
RT	000052	1:23	2:25	2:33	
TIME	000004	1:29	2:06		
TIT	000022	1:17	2:16		
TLOG	000042	1:21	2:11		
TYPE	000104	1:37	1:34		
UNIT	000023	1:18	1:30	2:02	2:25
UT	000124	1:25	2:15	2:38	
0ZAP	000006	1:15	2:34		
.130	000052	2:37	2:32		
.30	000047	2:36	1:36	2:13	
.LOG	000137	1:11	2:23		
.US0	000120	1:09	2:27		
.US3	000126	1:10	1:31	2:03	
.UT	000051	2:38	1:37		

FLAGGED LINES: NONE

TEN-76 UTILITY (TEN76)

The TEN-76 utility is used to assign a unit to a ticket. A ticket must be current (displayed on the terminal) and the unit must be both on duty and not assigned to another ticket. This utility is initiated by a call to the invoke function from the unit function. If a unit is already assigned to the ticket this unit is put into the assist field of the ticket. All relevant information in the ticket data block, the ticket status table entry and the unit status table entry are updated. The utility exits through the unit status output routine which displays the new status for this unit. Any information added to the ticket data block is also updated on the display.


```
                .HEAD   A I D S    T E N - 7 6    U T I L I T Y
                .TITLE  TEN76

                ;       AIDS TABLE ASSIGNMENTS

000102  .CVA=   102     ; ASCII CONVERSION
000103  .CVR=   103     ; BINARY CONVERSION
000104  TYPE=   104     ; TEXT STRING TYPER
000105  ASK=    105     ; DATA STRING INPUT
000106  .GET=   106     ; INPUT IN FORMAT
000120  .USO=   120     ; UNIT STATUS OUTPUT
000123  .TSI=   123     ; TICKET STATUS INSERT
000126  .US3=   126     ; UNIT STATUS LOCATE
000127  .TS3=   127     ; TICKET STATUS LOCATE
000137  .LOG=   137     ; MAKE LOG ENTRY

                ;       TIT WORD ASSIGNMENTS

000006  ZAP=    6       ; DISPLAY OUTPUT ENTRY
000014  FORM=   14      ; TICKET FORMAT BLOCK
000015  DATA=   15      ; TICKET DATA BLOCK
000016  FIELD=  16      ; FORMAT BLOCK NUMBER
000017  INDEX=  17      ; DATA CHARACTER INDEX
000020  CCN=    20      ; CONTROL CENTER NAME
000021  OPR=    21      ; OPERATOR NUMBER
000022  TIME=   22      ; CURRENT TIME OF DAY
000023  UNIT=   23      ; CURRENT UNIT NUMBER
000024  TKT=    24      ; CURRENT TICKET NUMBER

000040  CODE=   P0      ; STATUS CODE
000041  POST=   P1      ; POST NUMBER
000042  TLOG=   P2      ; TICKET NUMBER

000043  D5=     P3
000044  D4=     P4
000045  D3=     P5      ; DECIMAL DIGITS
000046  D2=     P6
000047  D1=     P7

000052  RT=     R2      ; RETURN ADDRESS

000064  OUT=    S4      ; DATA OUTPUT VECTOR

000060  COUNT=  S0      ; DATA BYTE COUNTER
000061  POINT=  S1      ; DATA BYTE POINTER

000062  SAVE=   S2      ; INCIDENT CODE

000120  UP=     120     ; POST NUMBER AREA
000124  UT=     124     ; STATUS INFO AREA
```

```
                ; TICKET DATA BLOCK ASSIGNMENTS

000004  REF=     4       ; TICKET NUMBER IN BLOCK

                ; FORMAT BLOCK FIELD INDICES

177776  FIC=     -2      ; INCIDENT CODE
177775  FIL=     -3      ; INCIDENT LOCATION
177774  FIP=     -4      ; INCIDENT POST
177773  FIT=     -5      ; INCIDENT TIME
177772  FAT=     -6      ; ASSIGNED TIME
177771  FDP=     -12     ; DISPATCHED BY
177770  FUN=     -13     ; UNIT NUMBER
177767  FOF=     -14     ; OFFICER(S)
177766  FAU=     -15     ; ASSIST UNITS
177765  FDT=     -17     ; DEPARTMENT
177763  FRP=     -22     ; FROM POST

                .DSECT

                ; UNIT ASSIGNED

000000  TEN76:  STA 3   RT,2     ; SAVE RETURN
                LDA 0   CCN,2    ; CONTROL CENTER
                LDA 1   UNIT,2   ; UNIT NUMBER
                JSR     @.US3    ; LOCATE IN TABLE
                JMP     ERROR    ; NOT IN TABLE

                LDA 1   5,3      ; TICKET NUMBER
                MOV#    1,1 SZR  ; IF NOT ZERO,
                JMP     ERROR    ; COMPLAIN

                LDA 1   .10      ; STATUS OFFSET
                ADDZL   3,1      ; STRING POINTER
                STA 1   POINT,2  ; SAVE FOR LATER

                LDA 3   DATA,2   ; DATA BLOCK
                LDA 1   REF,3    ; TICKET NUMBER
                MOV#    1,1 SZR  ; IF ZERO,
                JMP     ERROR    ; COMPLAIN
                STA 1   TLOG,2   ; SET FOR LOG

                ;       JSR     @.TS3    ; LOOK UP TICKET
                ;       SKP     0,0 SKP  ; NOT IN TABLE
                ;       LDA 0   7,3      ; GET TIME LIMIT
                ;       STA 0   TLIM,2   ; SAVE FOR LATER
```



```
                ; POST RESPONDING

000020 006104           JSR     @TYPE    ; TYPE "FROM"
000021 000311'          MSG
000022 024542           LDA 0   .4
000023 024504           LDA 1   .UP      ; POST OFFSET
000024 147000           ADD     2,1      ; MAKE ADDRESS
000025 006105           JSR     @ASK     ; GET POST NUMBER
000026 000401           NOP              ; IF NONE, OKAY

000027 021121           LDA 0   UP+1,2
000030 041047           STA 0   D1,2     ; DIGIT 1
000031 101300           MOVS    0,0
000032 041046           STA 0   D2,2     ; AND 2

000033 021124           LDA 0   UP,2
000034 041045           STA 0   D3,2     ; DIGIT 3
000035 101300           MOVS    0,0
000036 041044           STA 0   D4,2     ; AND 4

000037 124400           SUB     0,0
000040 041043           STA 0   D5,2     ; NO DIGIT 5
000041 006103           JSR     @.CVB    ; MAKE BINARY
000042 045044           STA 1   POST,2   ; POST NUMBER

                ; CHECK UNIT NUMBER

000043 025042           LDA 1   TLOG,2   ; TICKET NUMBER
000044 006503           JSR     @.SET    ; SETUP DATA BLOCK
000045 035014           LDA 3   FORM,2   ; TICKET FORMAT
000046 035407           LDA 3   T.SA,3   ; FORMAT START
000047 025760           LDA 1   FUN,3    ; UNIT FIELD
000050 006501           JSR     @.BYTL   ; GET DATA BYTE
000051 006103           JSR     @.CVB    ; MAKE BINARY
000052 125014           MOV#    1,1 SZR  ; IF NOT ZERO,
000053 000501           JMP     ASST     ; ASSISTING
```

```
                ; ASSIGNED UNIT

                        LDA 1   UNIT,2  ; UNIT NUMBER
                        JSR     @.CVA   ; MAKE ASCII
                        LDA 3   FORM,2  ; TICKET FORMAT
                        LDA 3   T.SA,3  ; FORMAT START
                        LDA 1   FUN,3   ; UNIT FIELD
                        JSR     @.PUSH  ; PUT DATA IN

                ; ADD OFFICERS NAMES

                        LDA 3   FORM,2  ; TICKET FORMAT
                        LDA 3   T.SA,3  ; FORMAT START
                        LDA 1   FON,3   ; OFFICER FIELD
                        STA 1   FIELD,2 ; SET INTO TIT
                        SUB     1,1
                        STA 1   INDEX,2 ; POSITION ZERO
                        LDA Ø   .3Ø     ; STRING LENGTH
                        STA Ø   COUNT,2 ; SET COUNTER

                ON:     LDA 3   POINT,2 ; GET POINTER
                        MOVZR   3,3     ; MAKE ADDRESS
                        LDA 1   Ø,3     ; GET DATA WORD
                        MOVS    1,1 SZC ; IF EVEN,
                        MOVS    1,1     ; SWAP BYTES
                        JSR     @OUT,2  ; OUTPUT DATA
                        ISZ     POINT,2 ; BUMP POINTER
                        DSZ     COUNT,2 ; AND COUNTER,
                        JMP     ON      ; AND REPEAT

                ; FROM POST. OPERATOR

                        LDA 1   POST,2  ; POST NUMBER
                        JSR     @.CVA   ; MAKE ASCII
                        LDA 3   FORM,2  ; TICKET FORMAT
                        LDA 3   T.SA,3  ; FORMAT START
                        LDA 1   FFP,3   ; FROM FIELD
                        JSR     @.PUSH  ; PUT DATA IN

                        LDA 1   OPR,2   ; OPERATOR
                        JSR     @.CVA   ; MAKE ASCII
                        LDA 3   FORM,2  ; TICKET FORMAT
                        LDA 3   T.SA,3  ; FORMAT START
                        LDA 1   FDB,3   ; DISPATCH FIELD
                        JSR     @.PUSH  ; PUT DATA IN

                ; ENTER TIME ASSIGNED

                        LDA 3   FORM,2  ; TICKET FORMAT
                        LDA 3   T.SA,3  ; FORMAT START
                        LDA 1   F25,3   ; ASSIGNED FIELD
                        JSR     @.TIME  ; INSERT TIME
                        JMP     ASSD    ; CONTINUE
```



```
00124'000004  .4:     4
00125'000010  .10:    10
00126'000030  .30:    30
00127'000120  .UP:    UP
00130'000124  .UT:    UT

              ; TYPE ERROR, RETURN

00131'046417  ERRCL:  JSR     @.RET   ; CLEAR TICKET
00132'024404  ERROR:  LDA 1   ERR     ; TYPE "?"
00133'020404          LDA Ø   .13Ø
00134'035052          LDA 3   RT,2    ; RETURN ADDRESS
00135'003006          JMP     @ZAP,2  ; EXIT THRU ZAP

00136'003477  ERR:    7*400+"?
00137'000130  .13Ø:   13Ø

              ; PRINT STATUS, RETURN

00140'006410  CLOUT:  JSR     @.RET   ; CLEAR TICKET
00141'021020          LDA Ø   CCN,2   ; CONTROL CENTER
00142'025023          LDA 1   UNIT,2  ; UNIT NUMBER
00143'035052          LDA 3   RT,2    ; RETURN ADDRESS
00144'002120          JMP     @.USØ   ; EXIT THRU STATUS

00145'000040  SPACE:  40
00146'000054  COMMA:  ",

              .EXTN   SET.,RET.,PULL.,PUSH.,TIME.
00147'177777  .SET:   SET.
00150'177777  .RET:   RET.
00151'177777  .PULL:  PULL.
00152'177777  .PUSH:  PUSH.
00153'177777  .TIME:  TIME.
```

```
                ; ASSISTING UNIT

                ASST:   LDA 3   FORM,2  ; TICKET FORMAT
                        LDA 3   T.SA,3  ; FORMAT START
                        LDA 1   FEX,3   ; ASSIST FIELD
                        STA 1   FIELD,2
                        SUB     1,1     ; POSITION ZERO
                        STA 1   INDEX,2

                PASS:   JSR     @.GET   ; GET A CHARACTER
                        LDA 0   SPACE   ; ASCII SPACE-CODE
                        SEQ     0,1     ; IF NOT A SPACE,
                        JMP     PASS    ; GO GET ANOTHER

                        LDA 1   COMMA
                        DSZ     INDEX,2 ; MOVE BACK ONE
                        STA     1,2     ; INSERT A COMMA
                        LDA 1   UNIT,2  ; UNIT NUMBER
                        JSR     @.CVA   ; MAKE ASCII
                        LDA 3   .PUSH
                        JSR     3,3     ; PUSH IT IN
```



```
                ; INCIDENT LOCATION

00175/035014    ASSD:   LDA 3   FORM,2  ; TICKET FORMAT
00176/035407            LDA 3   T.SA,3  ; FORMAT START
00177/025775            LDA 1   FIL,3   ; LOCATION FIELD
00200/045016            STA 1   FIELD,2 ; SET INTO TIT
00201/124004            SUB     0,0
00202/041017            STA 0   INDEX,2 ; POSITION ZERO
00203/020723            LDA 0   .30     ; STRING LENGTH
00204/041060            STA 0   COUNT,2 ; SET COUNTER
00205/024723            LDA 1   .UT     ; STRING OFFSET
00206/147120            ADDZL   2,1     ; BYTE POINTER
00207/045061            STA 1   POINT,2 ; SAVE FOR LATER

00210/006106    IL:     JSR     @.GET   ; GET A CHARACTER
00211/035061            LDA 3   POINT,2 ; PICK UP POINTER
00212/175224            MOVZR   3,3     ; MAKE ADDRESS
00213/021400            LDA 0   0,3     ; GET DATA WORD
00214/101342            MOVS    0,0 SZC ; SWAP; IF ODD,
00215/107000            ADD     0,1     ; COMBINE BYTES
00216/045400            STA 1   0,3     ; REPLACE DATA
00217/011061            ISZ     POINT,2 ; BUMP POINTER
00220/015064            DSZ     COUNT,2 ; AND COUNTER,
00221/000767            JMP     IL      ; AND REPEAT
00222/006137            JSR     @.LOG   ; MAKE LOG ENTRY

                ; UPDATE UNIT STATUS

00223/035014            LDA 3   FORM,2  ; TICKET FORMAT
00224/035407            LDA 3   T.SA,3  ; FORMAT START
00225/025774            LDA 1   FIP,3   ; POST FIELD
00226/006723            JSR     @.PULL  ; GET DATA OUT
00227/006103            JSR     .CVB    ; MAKE BINARY
00230/045041            STA 1   POST,2  ; POST NUMBER

00231/021024            LDA 0   CON,2   ; CONTROL CENTER
00232/025023            LDA 1   UNIT,2  ; UNIT NUMBER
00233/006126            JSR     @.USR   ; LOCATE IN TABLE
00234/000675            JMP     ERROL   ; CAN'T HAPPEN

00235/025040            LDA 1   CODE,2  ; GET THE STATUS
00236/045402            STA 1   2,3     ; INTO THE TABLE
00237/025022            LDA 1   TIME,2  ; CURRENT TIME
00240/045403            STA 1   3,3     ; INTO THE TABLE
00241/025041            LDA 1   POST,2  ; POST NUMBER
00242/045404            STA 1   4,3     ; INTO THE TABLE
00243/025024            LDA 1   TKT,2   ; TICKET NUMBER
00244/045405            STA 1   5,3     ; INTO THE TABLE
                ;       LDA 1   TLIM,2  ; TIMER LIMIT
                ;       STA 1   7,3     ; INTO THE TABLE
```

```
                ; UPDATE TICKET STATUS

                        LDA 3   FORM,2  ; TICKET FORMAT
                        LDA 3   T.SA,3  ; FORMAT START
                        LDA 1   FIC,3   ; CODE FIELD
                        JSR     @.PULL  ; GET DATA OUT
                        JSR     @.CVB   ; MAKE BINARY
                        STA 1   SAVE,2  ; INCIDENT CODE

                        LDA 3   FORM,2  ; TICKET FORMAT
                        LDA 3   T.SA,3  ; FORMAT START
                        LDA 1   FIT,3   ; TIME FIELD
                        JSR     @.PULL  ; GET DATA OUT
                        JSR     @.CVB   ; MAKE BINARY
                        STA 1   TIME,2  ; INCIDENT TIME

                        LDA 0   CCN,2   ; CONTROL CENTER
                        LDA 1   TKT,2   ; TICKET NUMBER
                        JSR     @.TSI   ; INSERT IN TABLE
                        JMP     CLOUT   ; NO ROOM IN TABLE

                        LDA 1   SAVE,2  ; INCIDENT CODE
                        STA 1   2,3     ; INTO THE TABLE
                        LDA 1   TIME,2  ; INCIDENT TIME
                        STA 1   3,3     ; INTO THE TABLE
                        LDA 1   POST,2  ; POST NUMBER
                        STA 1   4,3     ; INTO THE TABLE

                        LDA 1   5,3     ; UNIT NUMBER
                        MOVL#   1,1 SNC ; IF ZERO OR LESS,
                        LDA 1   UNIT,2  ; REPLACE IT
                        STA 1   5,3     ; INTO THE TABLE

                ; INCIDENT LOCATION

                        LDA 1   .30     ; STRING LENGTH
                        MOVR    1,1     ; MAKE WORD COUNT
OL:                     LDA 0   UT,2    ; OBTAIN A WORD
                        STA 0   24,3    ; AND INSERT IT
                        INC     3,3     ; BUMP ADDRESS
                        INC     2,2     ; THIS ONE TOO
                        INC     1,1 SZR ; BUMP COUNTER
                        JMP     OL      ; AND REPEAT

                        LDA 2   2       ; RECOVER TIT
                        JMP     CLOUT   ; AND RETURN

                .IXTX   1
                .IXTX   1

MSG:            .TXT    *   FROM*

                .END    TEN76
```


SYMBOL	VALUE	DEF'N	REFERENCES				
ASK	000105	1:10	3:08				
ASSD	000175'	7:03	4:53				
ASST	000154'	6:03	3:36				
CCN	000020	1:25	2:24	5:21	7:36	8:17	
CLOUT	000143'	5:20	8:20	8:46			
CODE	000040	1:31	7:41				
COMMA	000146'	5:27	6:15				
COUNT	000060	1:45	4:19	4:28	7:10	7:23	
D1	000047	1:39	3:12				
D2	000046	1:38	3:14				
D3	000045	1:37	3:17				
D4	000044	1:36	3:19				
D5	000043	1:35	3:22				
DATA	000015	1:22	2:37				
ERR	000136'	5:15	5:18				
ERRCL	000131'	5:09	7:39				
ERROR	000132'	5:10	2:27	2:31	2:40		
F25	177772	2:11	4:51				
FDB	177766	2:12	4:44				
FEX	177763	2:15	6:05				
FFP	177756	2:17	4:37				
FIC	177776	2:07	8:05				
FIELD	000016	1:23	4:15	6:06	7:06		
FIL	177775	2:08	7:05				
FIT	177774	2:09	7:51				
FI	177773	2:10	8:12				
FUN	177764	2:14	4:14				
FORM	000014	1:21	3:30	4:05	4:12	4:35	4:42
			7:03	7:29	8:03	8:10	
FPD	177761	2:16					
FUN	177765	2:13	3:32	4:07			
IL	000214'	7:15	7:24				
INDEX	000017	1:24	4:17	6:08	6:16	7:08	
MSG	000311'	8:51	3:[illegible]				
OL	000301'	8:38	8:43				
ON	000072'	4:21	4:29				
OPR	000021	1:26	4:10				
OUT	000064	1:43	4:26	6:17			
PASS	000162'	6:10	6:13				
POINT	000061	1:46	2:35	4:21	4:27	7:13	7:16
POST	000041	1:32	3:24	4:33	7:34	7:45	8:26
PULL.	'X	5:29	5:32				
PUSH.	'X	5:29	5:33				
RET	000004	2:03	2:33				
RET.	'X	5:29	5:31				
RT	000052	1:41	2:23	5:12	5:23		
SAVE	000062	1:48	8:08	8:22			
SET.	'X	5:29	5:30				
SPACE	000145'	5:26	6:11				
TEN76	000000'	2:23	8:53				
TIME	000022	1:27	7:43	8:15	8:24		
TKT.	'X	5:29	5:34				
TKT	000021	1:29	7:47	8:18			
TPOS	000042	1:33	2:41	3:28			
TYPE	000134	1:09	3:03				

/2 /..

...OL	VALU...	DEFN	REFERENCES					
I...	...423	1:28	2:25	4:03	5:22	6:18	7:37	8:31
	...123	1:50	3:11	3:16	5:04			
	...123	1:51	5:05	8:38				
	...[illegible]	1:2[illegible]	5:1[illegible]					
	...12...	5:02	2:33					
...	...13...	5:16	5:11					
...	...12...	5:03	4:1[illegible]	7:09	8:36			
...	...12...	5:01	3:05					
...A	...14	1:07	4:01	4:34	4:41	6:19		
...	...14	1:03	3:23	3:34	7:33	8:07	8:14	
...	...19...	1:11	6:1[illegible]	7:15				
...	...137	1:1[illegible]	7:25					
...	...15...	5:32	3:3[illegible]	7:32	8:06	8:13		
...	...15...	5:33	4:03	4:38	4:45	6:20		
...	...15...	5:31	5:07	5:20				
...	...14...	5:3[illegible]	3:29					
...	...15...	5:34	4:52					
...	...12...	1:1[illegible]	8:19					
...	...12...	1:15						
...	...[illegible]	[illegible]	3:[illegible]					
...	...	1:12	5:[illegible]					
...	...[illegible]	1:14	2:25	7:33				
...	...[illegible]	2:[illegible]	7:11					

...[illegible] : ...

TEN-22 UTILITY (TEN22)

The TEN-22 utility is used to cause a unit to disregard an assignment. The unit must be assigned to a ticket. The unit's status is set to 1ø-øø. All relevant information in the ticket data block, the ticket status table entry, the unit status table entry and on the displayed ticket is updated. The utility exits through the unit status output routine which displays the new status for this unit. This utility is initiated by a call to the invoke function from the unit function.


```
                .HEAD   A I D S   T E N - 2 2   U T I L I T Y

                .TITLE  TEN22

                ;       AIDS TABLE ASSIGNMENTS

000103          .CVB=   103     ; BINARY CONVERSION
000120          .US0=   120     ; UNIT STATUS OUTPUT
000126          .US3=   126     ; UNIT STATUS LOCATE
000127          .TS3=   127     ; TICKET STATUS LOCATE
000137          .LOG=   137     ; MAKE LOG ENTRY

                ;       TIT WORD ASSIGNMENTS

000006          ZAP=    6       ; DISPLAY OUTPUT ENTRY
000014          FORM=   14      ; TICKET FORMAT BLOCK
000016          FIELD=  16      ; FORMAT FIELD NUMBER
000017          INDEX=  17      ; DATA CHARACTER INDEX
000020          CCN=    20      ; CONTROL CENTER
000022          TIME=   22      ; CURRENT TIME OF DAY
000023          UNIT=   23      ; CURRENT UNIT NUMBER

000041          POST=   P1      ; POST NUMBER
000042          TLOG=   P2      ; TICKET NUMBER

000052          RT=     R2      ; RETURN ADDRESS
000051          RA=     R1      ; ALTERNATE RETURN

000064          OUT=    S4      ; DATA OUTPUT VECTOR

000063          COUNT=  S0      ; DATA BYTE COUNTER

000124          UI=     124     ; STATUS INFO AREA

                ;       FORMAT BLOCK FIELD INDICES

177765          FUN=    -13     ; UNIT FIELD INDEX
177764          FUN=    -14     ; NAME FIELD INDEX

                .NREL
```

```
                ; DISREGARD ASSIGNMENT

TEN22:  STA 3   RT,2    ; SAVE RETURN
        LDA 0   CCN,2   ; CONTROL CENTER
        LDA 1   UNIT,2  ; UNIT NUMBER
        JSR     @.US3   ; LOCATE IN TABLE
        JMP     ERROR   ; NOT IN TABLE
        LDA 0   6,3     ; PATROL AREA
        JSR     @.BEAT  ; LOOK IT UP

        LDA 0   CCN,2   ; CONTROL CENTER
        LDA 1   UNIT,2  ; UNIT NUMBER
        JSR     @.US3   ; LOCATE IN TABLE
        JMP     ERROR   ; WILL NOT HAPPEN

        LDA 1   4,3     ; POST NUMBER
        STA 1   POST,2  ; SET FOR LOG
        LDA 1   5,3     ; TICKET NUMBER
        STA 1   TLOG,2  ; SET FOR LOG

        LDA 0   TEN.8   ; STATUS IS 10-8
        STA 0   2,3     ; INTO THE TABLE
        LDA 0   TIME,2  ; CURRENT TIME
        STA 0   3,3     ; INTO THE TABLE

        SUB     0,0
        STA 0   4,3     ; CLEAR POST
        STA 0   5,3     ; AND TICKET

        LDA 1   .30     ; STRING LENGTH
        NEGOR   1,1     ; MAKE WORD COUNT
OL:     LDA 0   UT,2    ; OBTAIN A WORD
        STA 0   24,3    ; AND INSERT IT
        INC     3,3     ; BUMP ADDRESS
        INC     2,2     ; THIS ONE TOO
        INC     1,1 SZR ; BUMP COUNTER
        JMP     OL      ; AND REPEAT

        LDA 2   2       ; RECOVER TIT
        JSR     @.LOG   ; MAKE LOG ENTRY
```



```
00040'021020        LDA 0   CCN,2   ; CONTROL CENTER
00041'025042        LDA 1   TLOG,2  ; TICKET NUMBER
00042'125014        MOV#    1,1 SZR ; IF NOT ZERO,
00043'006127        JSR     @.TS3   ; LOCATE IN TABLE
00044'000446        JMP     STOUT   ; NOT IN TABLE

00045'025405        LDA 1   5,3     ; UNIT ON TICKET
00046'021023        LDA 0   UNIT,2  ; CURRENT UNIT
00047'106414        SEQ     0,1     ; IF NOT EQUAL,
00050'000442        JMP     STOUT   ; RETURN

00051'102400        SUB     0,0
00052'041405        STA 0   5,3     ; CLEAR UNIT
00053'025042        LDA 1   TLOG,2  ; TICKET NUMBER
00054'006455        JSR     @.SET   ; SETUP DATA BLOCK

00055'035014        LDA 3   FORM,2  ; TICKET FORMAT
00056'035407        LDA 3   T.SA,3  ; FORMAT START
00057'025765        LDA 1   FUN,3   ; UNIT FIELD
00060'006453        JSR     @.PULL  ; GET DATA OUT

00061'006103        JSR     @.CVB   ; MAKE BINARY
00062'021023        LDA 0   UNIT,2  ; CURRENT UNIT
00063'106411        SEQ     0,1     ; IF NOT EQUAL,
00064'000425        JMP     CLOUT   ; RETURN

                ; CLEAR UNIT FROM TICKET

00065'035014        LDA 3   FORM,2  ; TICKET FORMAT
00066'035407        LDA 3   T.SA,3  ; FORMAT START
00067'025765        LDA 1   FUN,3   ; UNIT FIELD
00070'020410        LDA 0   .4      ; SIZE = 4.
00071'004426        JSR     FILL    ; BLANK IT OUT

00072'035014        LDA 3   FORM,2  ; TICKET FORMAT
00073'035407        LDA 3   T.SA,3  ; FORMAT START
00074'025764        LDA 1   FUN,3   ; NAME FIELD
00075'020430        LDA 0   .30     ; SIZE = 24.
00076'004421        JSR     FILL    ; BLANK IT TOO

00077'000412        JMP     CLOUT   ; RETURN
```

```
                .4:     4
                .30:    30
                TTY.a:  1000.

                ; TYPE ERROR, RETURN

                ERR:    LDA 1   ERR     ; TYPE "?"
                        LDA 0   .130
                        LDA 3   RT,2    ; RETURN ADDRESS
                        JMP     @ZAP,2  ; EXIT THRU ZAP

                ERR:    7*400+"?
                .130:   130

                ; PRINT STATUS, RETURN

                CLOUT:  JSR     @.RET   ; CLEAR TICKET
                STOUT:  LDA 0   CC,2    ; CONTROL CENTER
                        LDA 1   UNIT,2  ; UNIT NUMBER
                        LDA 3   RT,2    ; RETURN ADDRESS
                        JMP     @.OUT   ; EXIT THRU STATUS

                SPACE:  40

                ; BLANK-FILL A FIELD

                FILL:   STA 3   RT,2    ; SAVE RETURN
                        STA 1   FIELD,2 ; SET FIELD NUMBER
                        STA 0   COUNT,2 ; AND BYTE COUNTER
                        SUB     0,0
                        STA 0   INDEX,2 ; POSITION ZERO

                JAM:    LDA 1   SPACE   ; ASCII SPACE
                        JSR     @OUT,2  ; STICK IT IN
                        ISZ     COUNT,2 ; BUMP COUNTER
                        JMP     JAM     ; AND REPEAT

                        JMP     @RT,2   ; RETURN

                .EXTN   SET.,RET.,PULL.,BEAT.
                .SET:   SET.
                .RET:   RET.
                .PULL:  PULL.
                .BEAT:  BEAT.

                .END    TEST22
```


SYMBOL VALUE DEF'N REFERENCES

```
BEAT.          'X  4:40   4:44
CON     000020     1:19   2:04   2:11   3:01   4:18
CLOUT   000111'    4:17   3:25   3:41
COUNT   000063     1:31   4:29   4:35
ERR     000107'    4:12   4:07
ERROR   000103'    4:07   2:07   2:14
FIELD   000016     1:17   4:28
FILL    000117'    4:27   3:33   3:39
FOO     177764     1:38   3:37
FOO1    000014     1:16   3:17   3:29   3:35
FUT     177765     1:37   3:19   3:31
INDEX   000017     1:18   4:31
JAM     000124'    4:33   4:36
OL      000130'    2:32   2:37
OUT     000064     1:29   4:34
POST    000001     1:23   2:17
PULL.          'X  4:43   4:43
RA      000051     1:27   4:27   4:38
RET.           'X  4:42   4:42
RT      000052     1:26   2:03   4:09   4:20
SET.           'X  4:40   4:41
SPACE   000116'    4:23   4:33
STOUT   000112'    4:18   3:05   3:10
TEST22  000000'    2:03   4:46
TTLOG   000132'    4:03   2:21
TTI     000022     1:20   2:23
TTLOG   000042     1:24   2:19   3:02   3:14
UNIT    000023     1:21   2:05   2:12   3:08   3:23   4:19
UT      000124     1:33   2:32
ZAP     000006     1:15   4:10
.130    000113'    4:13   4:08
.30     000131'    4:02   2:30   3:38
.4      000004'    4:01   3:32
.BEAT   000134'    4:44   2:09
.CVB    000103     1:17   3:22
.LOG    000137     1:11   2:40
.PULL   000133'    4:43   3:20
.RET    000132'    4:42   4:17
.SET    000131'    4:41   3:15
.TSB    000127     1:10   3:04
.USR    000121     1:06   4:21
.USR    000126     1:12   2:09   2:13
```

FLAGGED LINES: NONE

TICKET BLOCK SETUP (SETUP)

Ticket block setup contains two routines:

SET.: Saves vital parameters of current ticket while calling up a new ticket.

RET.: Restores previous ticket to current status while current ticket becomes previous ticket.


```
                .HEAD   A I D S    T I C K E T    B L O C K    S E T U P

                .TITLE  SETUP

                ;       AIDS TABLE ASSIGNMENTS

000107          .PUT=   107     ; OUTPUT IN FORMAT
000114          .DATA=  114     ; DISPLAY IN FORMAT
000132          .FIND=  132     ; LOCATE TICKET BLOCK

                ;       TIT WORD ASSIGNMENTS

000014          FORM=   14      ; TICKET FORMAT BLOCK
000015          DATA=   15      ; TICKET DATA BLOCK
000024          TKT=    24      ; CURRENT TICKET NUMBER
000025          OLD=    25      ; PREVIOUS TICKET NUMBER

000001          RA=     R1      ; RETURN ADDRESS

000004          OUT=    S4      ; DATA OUTPUT VECTOR
000005          SAV1=   S5      ; TICKET NUMBER STORAGE
000006          SAV2=   S6      ; FORMAT BLOCK STORAGE
000007          SAV3=   S7      ; DATA BLOCK STORAGE

                ; TICKET DATA BLOCK ASSIGNMENTS

000007          MOD=    7       ; DATA BLOCK UPDATE FLAG
000006          USE=    6       ; DATA BLOCK USAGE COUNT
000005          BLK=    5       ; BLOCK NUMBER IN TICKET
000004          REF=    4       ; TICKET NUMBER IN BLOCK

                .ENT    SET.,RET.
                .REL
```

```
                .4%:    40
                .12%:   124
                .13%:   134
                .PI:    PI

                .TXTV   1
                .TXTX   1

                %%:     7*40+"?
                SG:     .TXT    *CONFIRM: FILE SEQUENCE*

                ; TYPE ERROR MESSAGE

                QZAP:   JSR     @.RET   ; RELEASE TICKET
                ERR0:   LDA 1   ERR     ; TYPE "?"
                        LDA 0   .134
                        LDA 3   RT.2    ; RETURN ADDRESS
                        JMP     @ZAP.2  ; EXIT THRU ZAP

                .EXTN   SET..RET.
                .SET:   SET.
                .PSE:   PSE.

                .       FILE.
```


SYMBOL	VALUE	DEF'N	REFERENCES
ASK	000105	1:09	2:13
CC!	000020	1:21	2:23 2:24
D1	000047	1:27	3:31
D2	000046	1:26	3:29
D3	000045	1:25	3:27
D4	000044	1:24	3:25
D5	000043	1:23	3:23
DATA	000115	1:15	2:30
ER%	000106'	4:09	4:15
ERROR	000124'	4:15	2:26 2:37
FIL	177763	1:43	2:42
FILD	000116	1:19	2:43
FILE.	000134'	2:43	4:24
FN	000113	1:35	2:18 4:04
FN%1	000114	1:17	2:40
GET	000115'	2:20	2:14
GOT	000121'	2:25	2:19
INDEX	000117	1:20	2:45 2:53
LP	000137'	4:13	2:09
OUT	000064	1:31	3:06 3:11 3:14 3:24 3:26 3:29 3:35
			3:32
OUTT	000123'	4:14	2:52
PAT	000004	1:39	2:35
PSE.	'X	4:20	4:22
PT	000052	1:29	2:03 3:35 4:17
PX	000050	1:33	2:27 3:03 3:08 3:12 3:18
QAP.	'X	4:20	4:21
READ	000141	1:12	2:33
TAPE	000104	1:08	2:16
ZAP	000060	1:16	2:46 2:22 2:30 4:18
.120	000120'	4:12	2:21
.134	000134'	4:13	2:29 4:16
.EX	000102'	4:21	2:51
.IOX	000131	1:11	2:26
.MA	000102	1:07	3:21
.PE	000105'	4:04	2:11
.RET	000136	1:10	2:49
.SET	000131'	4:22	3:34 4:14
.SFT	000134'	4:21	2:38

FLAGGED LINES: NONE

```
                ; GET CONFIRMATION

                FILE.:  STA 3   RT,2    ; SAVE RETURN
                        SUB     0,0
                        LDA 1   TAD     ; POSITION
                        JSR     @ZAP,2

                        JSR     @TYPE   ; TYPE "CONFIRM"
                        MSG
                        SUBZL   0,0     ; = 1
                        LDA 1   .FN     ; DATA OFFSET
                        ADD     2,1     ; MAKE ADDRESS
                        JSR     @ASK    ; CONFIRMATION
                        JMP     GET     ; USE DEFAULT

                ; LOOK UP FILE SEQUENCE

                        LDA 0   FN,2    ; GET INPUT DATA
                        JMP     GOT     ; JUMP IN LATER
                GET:    LDA 1   CCN,2   ; DEFAULT CCN
                        LDA 0   .120
                        JSR     @ZAP,2  ; DISPLAY IT

                        LDA 0   CCN,2   ; DEFAULT CCN
                GOT:    JSR     @.CCN   ; LOCATE IN TABLE
                        JMP     ERROR   ; NOT IN TABLE
                        STA 3   SAVE,2  ; SAVE TABLE ENTRY
                        LDA 1   2,3     ; FILE SEQUENCE
                        LDA 0   .130
                        JSR     @ZAP,2  ; DISPLAY IT

                ; SET UP TICKET BLOCK

                        LDA 3   DATA,2  ; DATA BLOCK
                        LDA 1   REF,3   ; TICKET NUMBER
                        MOVZ    1,1 SNR ; IF ZERO.
                        JMP     ERROR   ; COMPLAIN
                        JSR     @.SET   ; SETUP DATA BLOCK

                        LDA 3   FORM,2  ; TICKET FORMAT
                        LDA 3   T.S4,3  ; FORMAT START
                        LDA 1   FFN,3   ; FILE FIELD
                        STA 1   FIELD,2 ; INSTALL IN TIT
                        SUB     0,0
                        STA 0   INDEX,2 ; POSITION ZERO

                ; LOOK FOR FILE NUMBER

                        JSR     @.GET   ; GET A CHARACTER
                        LDA     .SP     ; ASCII SPACE
                        SNA     1,0 SZR ; IF EQUAL,
                        JMP     QUIT    ; DO NOTHING
                        STA     INDEX,2 ; BACK UP AGAIN
```



```
                ; INSERT FILE SEQUENCE

00047/035060            LDA 3   SAVE,2  ; ENTRY ADDRESS
00050/025401            LDA 1   0,3     ; CONTROL CENTER
00051/125300            MOVS    1,1     ; MOVE TO LOW BYTE
00052/007064            JSR     @OUT,2  ; INSERT IN TICKET

00053/035060            LDA 3   SAVE,2  ; ENTRY ADDRESS
00054/025402            LDA 1   2,3     ; FILE SEQUENCE
00055/125300            MOVS    1,1     ; INSERT HI-BYTE
00056/007064            JSR     @OUT,2
00057/035060            LDA 3   SAVE,2  ; ENTRY ADDRESS
00060/025402            LDA 1   2,3     ; FILE SEQUENCE
00061/007064            JSR     @OUT,2  ; INSERT LO-BYTE

                ; INSERT FILE NUMBER

00062/035060            LDA 3   SAVE,2  ; ENTRY ADDRESS
00063/011403            ISZ     3,3     ; BUMP, PICK UP
00064/025403            LDA 1   3,3     ; FILE NUMBER
00065/006102            JSR     @.CVA   ; MAKE ASCII

00066/025043            LDA 1   D5,2    ; DIGIT 5
00067/007064            JSR     @OUT,2
00070/025044            LDA 1   D4,2    ; DIGIT 4
00071/007064            JSR     @OUT,2
00072/025045            LDA 1   D3,2    ; DIGIT 3
00073/007064            JSR     @OUT,2
00074/025046            LDA 1   D2,2    ; DIGIT 2
00075/007064            JSR     @OUT,2
00076/025047            LDA 1   D1,2    ; DIGIT 1
00077/007064            JSR     @OUT,2

00100/006431            JSR     @.RET   ; RELEASE TICKET
00101/003052            JMP     @RT,2   ; AND RETURN
```

FILE UTILITY (FILE.)

The file utility assigns a file sequence number to the current ticket. A ticket must be current and the control center name must be valid. The message "CONFIRM: FILE SEQUENCE" is displayed and the operator may type in a control center name. If nothing is typed in, the utility will use the name entered for the terminal. The file number is added to the ticket data block and the displayed ticket, and the control center table entry is updated.


```
                .HEAD   A I D S    F I L E    U T I L I T Y

                .TITLE  FILE.

                ;       AIDS TABLE ASSIGNMENTS

000102  .CVA=    102     ; ASCII CONVERSION
000104  TYPE=    104     ; AIDS TEXT TYPER
000105  ASK=     105     ; AIDS DATA INPUT
000106  .GET=    106     ; INPUT IN FORMAT
000130  .CCN=    130     ; CENTER LOOKUP
000140  TAD=     140     ; TAG-LINE ADDRESS

                ;       TIT WORD ASSIGNMENTS

000006  ZAP=     6       ; DISPLAY COMMAND ENTRY
000014  FORM=    14      ; FORMAT BLOCK ADDRESS
000015  DATA=    15      ; DATA BLOCK ADDRESS
000016  FIELD=   16      ; FORMAT FIELD NUMBER
000017  INDEX=   17      ; FIELD POSITION INDEX
000020  CCN=     20      ; CONTROL CENTER

000043  D5=      P3
000044  D4=      P4
000045  D3=      P5      ; DECIMAL DIGITS
000046  D2=      P6
000047  D1=      P7

000052  RT=      R2      ; RETURN ADDRESS

000064  OUT=     S4      ; DATA OUTPUT VECTOR

000063  SAVE=    S0      ; SAVED TABLE ENTRY

000110  FN=      110     ; DATA OFFSET

                ;       DATA BLOCK ASSIGNMENTS

000004  REF=     4       ; TICKET NUMBER IN BLOCK

                ;       FORMAT FIELD INDICES

177760  FFN=     -20     ; FILE NUMBER FIELD INDEX

                .NREL
```

[illegible]	DEFN	REFERENCES
[illegible]	1:12	1:[illegible]2
[illegible]	1:[illegible]7	1:12
[illegible]SX	1:[illegible]5	1:13

[illegible] LIBRARIES: NONE

FILE FUNCTION (FILE.)

The file function calls the invoke function .CALL to initiate the file utility FILE. .

```
PAGE  1        A I D S    F I L E    F U N C T I O N
02/20/74
                        .HEAD   A I D S    F I L E    F U N C T I O N
                        .TITLE  FILE.
                        .EXTD   .CALL   ; INVOKE ENTRY
          000050        RC=     R0      ; RETURN ADDRESS
                        .ENT    FILE.
                        .NREL
000000'055050   FILE.:  STA 3   RC,2    ; SAVE RETURN
000001'006001$          JSR     @.CALL  ; CALL FUNCTION
000002'000003'          .+1
          000001        .TXTM   1
000003'040511   .TXT    *AIDS.FILE*
000004'042123
000005'042706
000006'044514
000007'042400
                        .END
```

```
                        ; GET DATA FROM FIELD

                PULL.:  STA 1   FIELD,2 ; FIELD NUMBER
                        SUB     1,1
                        STA 1   INDEX,2 ; POSITION ZERO
                        STA 3   RA,2    ; SAVE RETURN

                        STA 1   D5,2    ; DIGIT 5
                        JSR     =,GET
                        STA 1   D4,2    ; DIGIT 4
                        JSR     =,GET
                        STA 1   D3,2    ; DIGIT 3
                        JSR     =,GET
                        STA 1   D2,2    ; DIGIT 2
                        JSR     =,GET
                        STA 1   D1,2    ; DIGIT 1

                        JMP     @RA,2   ; RETURN

                        ; PUT DATA INTO FIELD

                PUSH.:  STA 1   FIELD,2 ; FIELD NUMBER
                        SUB     1,1
                        STA 1   INDEX,2 ; POSITION ZERO
                        STA 3   RA,2    ; SAVE RETURN

                        LDA 1   D4,2    ; DIGIT 4
                        JSR     SEND
                        LDA 1   D3,2    ; DIGIT 3
                        JSR     SEND
                        LDA 1   D2,2    ; DIGIT 2
                        JSR     SEND
                        LDA 1   D1,2    ; DIGIT 1
                        JSR     SEND

                        JMP     @RA,2   ; RETURN

                SEND:   LDA 0   D5,2
                        SUB     0,1
                        JMP     3,3
                        DSZ     D5,2
                        JMP     @JT,2

                        .END
```



```
SYMBOL  VALUE   DEFN    REFERENCES

D1      000047  1:17    2:16    2:33
D2      000046  1:16    2:14    2:31
D3      000045  1:15    2:12    2:29
D4      000041  1:14    1:32    2:10    2:27
D5      000043  1:13    1:39    2:08    2:38    2:41
FIELD   000016  1:08    2:03    2:22
INDEX   000017  1:09    2:05    2:24
JMP     000004  1:21    2:12
PULL.   000015' 2:13    1:23    1:31
PUSH.   000033' 2:22    1:23    1:40
RA      000051  1:19    2:06    2:18    2:25    2:36
SEND    000062' 1:26    1:30    1:35    1:41
SEND    000062' 2:28    2:29    2:31    2:32    2:34
GET     000114' 1:43    1:33
INDEX   000022  1:11    1:37
PULL.   000015' 1:23    1:23
PUSH.   000033' 1:40    1:23
RA      000051  1:45    1:36
RA      000051  1:46    2:09    2:11    2:13    2:15

        FLAGGED LINES:  NONE
```

PUSH/PULL MODULE (MOVE.)

The push/pull module contains three small routines:

PUSH.: Takes four ASCII digits from the parameter words, and inserts them into the current data field.

PULL.: Takes four ASCII digits from the current data field, and inserts them into the parameter words.

TIME: If the first character position of the current field is blank, the field is filled with the current time.


```
                        .HEAD   A I D S     P U S H / P U L L       M O D U L E

                        .TITLE  MOVE.

          000102        .CVA=   102     ; ASCII CONVERSION
          000106        .GET=   106     ; INPUT IN FORMAT

          000016        FIELD=  16      ; FORMAT FIELD NUMBER
          000017        INDEX=  17      ; DATA CHARACTER INDEX

          000022        TIME=   22      ; CURRENT TIME

          000043        D5=     P3
          000044        D4=     P4
          000045        D3=     P5      ; DECIMAL DIGITS
          000046        D2=     P6
          000047        D1=     P7

          000051        RA=     R1      ; RETURN ADDRESS

          000064        OUT=    S4      ; DATA OUTPUT VECTOR

                        .ENT    PULL..PUSH..TIME.
                        .NREL

    00000'000000        RET:    0

                        ; INSERT A NEW TIME

    00001'054777        TIME.:  STA 3   RET     ; SAVE RETURN
    00002'004413                JSR     PULL.   ; GET OLD TIME
    00003'025044                LDA 1   D4,2    ; FIRST DIGIT
    00004'020410                LDA 0   SPACE   ; ASCII SPACE
    00005'106414                SEQ     0,1     ; IF BLANK,
    00006'002772                JMP     @RET    ; RETURN
                                
    00007'025022                LDA 1   TIME,2  ; CURRENT TIME
    00010'006102                JSR     @.CVA   ; MAKE ASCII
    00011'015043                DSZ     D5,2    ; DON'T SUPRESS
    00012'004422                JSR     PUSH.+1 ; PUT NEW TIME
    00013'002765                JMP     @RET    ; AND RETURN

    00014'000040        SPACE:  40
```

A I D S P A T R O L A R E A L O O K U P

...CE	DEF'N	REFERENCES					
124	1:15	2:44					
11	1:02	1:35	2:16	2:27			
M	1:37	1:26					
	1:14	1:54	2:07	4:38	5:01	5:05	
261	1:13	2:03	4:31	4:55			
120	1:17	1:37					
262'	2:24	1:39	1:50				
143'	3:28	3:46					
1 1'	2:21	1:42	3:07				
117'	3:13	2:13					
124'	3:10	4:26					
135'	3:21	3:31					
264'	5:14	4:13	4:33				
163'	3:18	3:43	4:07	4:11			
213'	4:19	3:56					
236'	4:21	3:03					
136'	2:18	4:45					
223'	4:18	3:17	3:21	3:36	3:51	4:19	
229'	4:24	4:14					
52	1:11	1:31	2:28				
157'	2:30	2:12	2:42				
271'	5:15	3:12	3:23	3:42			
16	1:26	4:24	4:39	4:41	5:02	5:15	5:24
211'	4:16	3:12	3:30				
12	1:12	4:17					
10	1:20	2:41	2:32	2:41	3:04	4:12	4:25
		5:23					5:16
15	1:25	2:39	2:40	3:03	5:06		
11	1:21	3:05	4:24				
14	1:24	3:11	3:17	3:22	3:28	3:37	
13	1:23	2:17	2:14	3:49	4:01	4:06	4:16
263'	5:18	4:09	4:20				
264'	5:19	2:37	4:35	5:21			
175'	2:45	2:08					
265'	5:13	4:51					
176'	2:46	2:31	3:26	3:32			
246'	5:11	3:13	3:24	3:43	3:52		
267'	5:12	3:53					
171'	2:14	2:06					
132	1:15	1:33					
177'	2:17	1:56					
111'	2:14	1:45	1:53	2:18			
7''	5:13	3:38					

```
                        LDA 2   BSTR,2  ; SET NEW BASE REGISTER
                        JMP     @SECR,2 ; RETURN

                FXIT:   LDA 2   2,0     ; RESET BASE REGISTER
                        LDA 2   BSTR,2  ; SET NEW BASE REGISTER
                        JMP     @SRET,2 ; RETURN

                .12:    12              ; ASCII LINE FEED
                .177:   177             ; 7-BIT MASK
                .377:   377             ; 8-BIT MASK
                .60:    60              ; ASCII ZERO
                .71:    71              ; ASCII NINE
                .STAR:  "*              ; ASCII STAR "*"

                SCHR:   STA 3   SECR,2  ; SAVE RETURN
                        LDA 3   SPTR,2  ; GET STRING POINTER
                        MOVZR   3,3     ; CONVERT TO WORD ADDRESS
                        LDA 0   0,3     ; GET DATA WORD
                        MOVS    0,0 SZC ; IF EVEN,
                        MOVS    0,0     ; JUMP ALREADY THERE
                        LDA 1   .177    ; READY MASK
                        AND     1,0     ; EXTRACT CHARACTER
                        ISZ     SPTR,2  ; BUMP STRING POINTER
                        JMP     @SECR,2 ; RETURN

                .END
```

```
                LDA 0   .30     ; DISPLACEMENT
                ADD     0,3     ; DISPLACE INTO BUFFER
                STA 3   BUF,2   ; SET UP DISK BUFFER ADDRESS

                LDA 3   .AN     ; GET OFFSET
                ADDZL   2,3     ; CREATE POINTER
                LDA 2   BSTR,2  ; SET UP BASE REGISTER
                STA 3   SPTR,2  ; INITIALIZE STRING POINTER
                SUB     3,3     ; CREATE A ZERO
                STA 3   TPTR,2  ; INITIALIZE TABLE POINTER

                JSR     SCAN    ; GET FIRST CHARACTER
                JSR     FIND    ; FIND A NAME
                LDA 3   TPTR,2  ; GET TABLE POINTER
                LDA 2   2,1     ; RESTORE AC2
                STA 3   AREA,2  ; RETURN RESULT

QUIT:           LDA 2   .IOCB   ; GET IOCB ADDRESS
                SUBZL   0,0     ; CREATE A +1
                STA 0   1,2     ; SET FUNCTION TO DEALLOCATE
                .IO             ; DEALLOCATE DISK BUFFER
                ABORT           ; MUST NOT FAIL

                LDA 2   2,3     ; RESTORE BASE REGISTER
                SUBZL   0,0     ; CREATE A +1
                .IO             ; GIVE UP THE DISK
                LDA 3   AREA,2  ; PICK UP RESULT
                JMP     @RET,2  ; RETURN

SCAN:           STA 3   SRET,2  ; SAVE RETURN ADDRESS
                LDA 1   .AS     ; ASCII SPACE
                LDA 3   PTR,2   ; GET STRING POINTER
                MOVZL   3,3     ; CONVERT TO WORD ADDRESS
                LDA 0   0,3     ; GET DATA WORD
                MOVS    0,1,SZC ; IF EVEN,
                MOVS    0,1     ; SWAP BYTES
                LDA 3   .377    ; FETCH MASK
                AND     3,1     ; EXTRACT CHARACTER
                SUB     0,1     ; SPACE?
                JMP     @SRET,2 ; NO, RETURN
                INC     SPTR,2  ; YES, BUMP POINTER
                JMP     SCAN+2  ; TRY THIS CHARACTER

.AN:            AN              ; AREA NAME DISPLACEMENT
.B:             B               ; BUFFER DISPLACEMENT
.AS:            AS              ; ASCII SPACE
.BUF:           BUFR            ; BUFFER DISPLACEMENT

.IOCB:  .BLK    !
        .BYTE   !
        .ASCII  !
.FIB:   .+1
        .TXT    "XXX.PATROL.AREAS*

.IOCB:  .+1                     ; I/O CONTROL BLOCK
        .BLK    1               ; CHANNEL
        .BLK    1               ; FUNCTION
        .BLK    1               ; TYPE / BUFFER ADDRESS
        .BLK    4               ; NOT USED
```



```
02/20/74
  1  00116/000470          470             ; SIZE OF BUFFER
  2
  3  00117/056016 FIND:    STA 3   SRET,2  ; SAVE RETURN ADDRESS
  4  00120/035011          LDA 3   SPTR,2  ; GET INITIAL VALUE OF STRING POINTER
  5  00121/055011          STA 3   STMP,2  ; SAVE IT
  6
  7  00122/024756          LDA 1   FILE    ; GET FILE NAME POINTER
  8  00123/004413          JSR     PRIME   ; PRIME INPUT BUFFER
  9
 10  00124/004477 FIND1:   JSR     RDCH    ; READ A FILE CHARACTER
 11  00125/041014          STA 0   TEMP,2  ; SAVE IT
 12  00126/004543          JSR     SCHR    ; GET NEXT STRING CHARACTER
 13  00127/024537          LDA 1   .60     ; ASCII ZERO
 14  00130/106113          SGE     0,1     ; ALPHA?
 15  00131/000775          JMP     .-3     ; YES, TRY NEXT ONE
 16
 17  00132/035014          LDA 1   TEMP,2  ; PICK UP FILE CHARACTER
 18  00133/106414          SEQ     0,1     ; SAME?
 19  00134/000455          JMP     SPEN    ; NO, TRY NEXT FILE ENTRY
 20
 21  00135/004466 FNXT:    JSR     RDCH    ; READ A FILE CHARACTER
 22  00136/041014          STA 0   TEMP,2  ; SAVE IT
 23  00137/004532          JSR     SCHR    ; GET NEXT STRING CHARACTER
 24  00140/024526          LDA 1   .60     ; ASCII ZERO
 25  00141/106113          SGE     0,1     ; ALPHA?
 26  00142/020734          LDA 0   .40     ; NO, MAKE SPACE
 27
 28  00143/035014 ENXT:    LDA 1   TEMP,2  ; AND FILE CHARACTER
 29  00144/106414          SEQ     0,1     ; SAME?
 30  00145/000443          JMP     SPEN    ; NO, TRY NEXT FILE ENTRY
 31
 32  00146/024733          LDA 1   .41     ; ASCII SPACE
 33  00147/106414          SEQ     0,1     ; SPACE?
 34  00150/000765          JMP     FNXT    ; NO, COMPARE NEXT PAIR
 35
 36  00151/004452          JSR     RDCH    ; READ NEXT FILE CHARACTER
 37  00152/041014          STA 0   TEMP,2  ; SAVE IT
 38  00153/024515          LDA 1   .STAR   ; ASCII "*"
 39  00154/106415          SEQ     0,1     ; STAR?
 40  00155/000406          JMP     ISTP    ; YES, CONTINUE
 41
 42  00156/004513          JSR     SCHR    ; GET NEXT STRING CHARACTER
 43  00157/024507          LDA 1   .60     ; ASCII ZERO
 44  00160/106113          SGE     0,1     ; ALPHA?
 45  00161/000775          JMP     .-3     ; NO, TRY NEXT ONE
 46  00162/000761          JMP     FNXT    ; COMPARE NOW
 47
 48  00163/176400 ISTP:    SUB     3,3     ; CREATE A ZERO
 49  00164/055013          STA 3   TPTR,2  ; SET TABLE POINTER = 0
 50
 51  00165/004436          JSR     RDCH    ; READ NEXT FILE CHARACTER
 52  00166/024541          LDA 1   .60     ; ASCII ZERO
 53  00167/034540          LDA 3   .71     ; ASCII NINE
 54  00170/113613          SUBZ    3,3     ; >9 ?
 55  00171/106113          SGE     0,1     ; >= 0 ?
 56  00172/000411          JMP     NOTD    ; NOT A DIGIT
 57
 58  00173/122440          SUB     1,0     ; CONVERT TO BINARY
```

PATROL AREA LOOKUP (AREA.)

The patrol area lookup routine receives as input an ASCII string in the terminal information table. It searches through the file "XX.PATROL.AREAS" looking for an entry with a matching string. If one is found, the corresponding beat number is read from the file and stored in a table parameter word. If anything goes wrong, the parameter word is set to zero.

The string parsing and matching technique is the same one used by the location lookup routine.

```
PAGE  1        A I D S    P A T R O L    A R E A    L O O K U P
02/20/74
                .HEAD   A I D S    P A T R O L    A R E A    L O O K U
                .TITLE  AREA.

          000130 .CCN=   130     ; CONTROL CENTER LOOKUP

          000020 CCN=    20      ; CONTROL CENTER NAME

          000041 AREA=   P1      ; PATROL AREA NUMBER

          000052 RTRN=   R2      ; MAIN PROGRAM RETURN ADDRESS

          000061 BUF=    S1      ; DISK BUFFER HEADER ADDRESS
          000062 BSTR=   S2      ; BLOCK STORAGE BASE ADDRESS

          000124 AT=     124     ; STRING OFFSET

                 ;       BLOCK STORAGE DISPLACEMENTS

          000010 SPTR=   10      ; STRING POINTER
          000011 STMP=   11      ; TEMPORARY STRING POINTER
          000012 SPIN=   12      ; INITIAL STRING POINTER
          000013 TPTR=   13      ; TABLE POINTER
          000014 TEMP=   14      ; MISC. TEMPORARY STORAGE
          000015 SRET=   15      ; SUBROUTINE RETURN ADDRESS
          000016 SECR=   16      ; SECONDARY SUBROUTINE RETURN ADDRESS

                 .EXIT   AREA.
                 .PAGE

000000'055052  AREA.:  STA 3   RTRN,2  ; SAVE RETURN
000001'132524          SUBZL   0,0     ; CREATE A +1
000002'004450          ENQ             ; GRAB THE DISK
000003'102400          SUB     0,0     ; CREATE A ZERO
000004'041041          STA 0   AREA,2  ; SET RESULT TO ZERO

000005'021020          LDA 0   CCN,2   ; CONTROL CENTER
000006'000130          JSR     @.CCN   ; LOCATE IN TABLE
000007'000443          JMP     DONE    ; NOT IN TABLE

000010'025401          LDA 1   1,3     ; GET CEN PT NAME
000011'046467          STA 1   @FILE   ; CREATE FILE NAME

000012'102400          SUB     0,0     ; CREATE A ZERO
000013'030476          LDA 2   .IOCB   ; SET IOC ADDRESS
000014'041001          STA 0   1,2     ; SET FUNCTION TO ALLOCATE
000015'101520          INCZL   0,0     ; CREATE A +2
000016'041002          STA 0   2,2     ; SET TYPE TO DISK FILE
000017'000004          .IO             ; ALLOCATE DISK SPACE
000020'000432          JMP     DONE    ; EXIT IF UNABLE TO ALLOCATE

000021'026432          LDA 2   2,0     ; RESTORE IOCB
000022'030447          LDA 3   .IOCB   ; GET IOCB ADDRESS
000023'032002          LDA 3   2,3     ; GET DISK FILE NAME
000024'020453          LDA 0   .FNB    ; DISPLACEMENT
000025'117000          ADD     0,3     ; DISPLACE TO ADDRESS

000026'055062          STA 3   BSTR,2  ; SET UP STORAGE ADDRESS
```

```
                SRET:   0       ; SUBROUTINE RETURN ADDRESS
                SPTR:   0       ; STRING POINTER
                SCTR:   0       ; STRING COUNTER
                VALUE:  0       ; EXTRACTED VALUE FROM FILE

                .STAR:  "*      ; ASCII "*"
                ..0:    "0      ; ASCII ZERO
                ..9:    "9      ; ASCII NINE
                .40:    40      ; ASCII SPACE
                .CR:    15      ; ASCII RETURN
                .LF:    12      ; ASCII LINE-FEED
                .AN:    AN      ; STRING OFFSET
                .24.:   24.     ; STRING LENGTH
                .177:   177     ; DATA BYTE MASK
                .377:   377     ; HALF-WORD MASK

                .TXTM   1
                .BLK    1

                FILE:   .+1     ; FILE NAME
                .TXT    *XX.PATROL.AREAS*
                BUFF:   .+1     ; DISK BUFFER
                .BLK    B.LEN

                .END
```


SYMBOL	VALUE	DEF'N	REFERENCES			
AN	000124	1:13	4:12			
BEAT.	000000'	1:18	1:15			
BUFF	000177'	4:22	1:44	3:04		
CCH	000020	1:07	1:25			
DONE	000066'	2:38	2:49			
ERROR	000076'	2:47	1:27	1:33	3:07	
FILE	000166'	4:20	1:29			
FILL	000070'	2:40	2:45			
GET	000101'	3:03	2:03	2:15		
GOTIT	000062'	2:33	2:17			
ISTAR	000040'	2:12	2:08			
PUT	000127'	3:34	2:32	2:44		
RTRN	000051	1:09	1:13	2:42		
SCAND	000042'	2:15	2:22	2:31		
SCAN3	000031'	2:03	2:05	2:10		
SCTR	000152'	4:03	2:40	3:29	3:34	3:39
SETUP	000121'	3:25	2:01	2:43		
SPTR	000151'	4:02	3:27	3:38	3:40	
TEST	000153'	4:04	3:13	3:11	3:37	3:52
START	000027'	2:01	2:36			
TCRLF	000112'	3:15	2:04	2:16		
TGT	000062	1:11	1:19	2:34		
VALUE	000153'	4:04	2:13	2:25	2:30	2:33
.177	000164'	4:14	3:33			
.24	000163'	4:13	3:28			
.3	000165'	4:15	1:41	3:42		
.4	000157'	4:09	2:43			
.AN	000162'	4:12	3:25			
.CCH	000130	1:05	1:26			
.CR	000132'	4:10	3:15			
.LF	000161'	4:11	3:13			
.STAR	000151'	4:06	2:25			
..0	000150'	4:07	2:18			
..9	000156'	4:08	2:19			

FLAGGED LINES: NONE

```
                        ; INITIALIZE STRING POINTER & COUNTER
                START:  JSR     SETUP
                        ; READ A CHARACTER
                SCANS:  JSR     GET
                        ; IS IT A "CR" OR "LF" ?
                        JSR     TCRLF
                        ; YES, IGNORE IT
                        JMP     SCANS
                        ; ASCII "*"
                        LDA 1   .STAR
                        ; IS IT A "*" ?
                        SNE     0,1
                        ; YES, GET DIGITS
                        JMP     ISTAR
                        ; NO, PUT INTO STRING
                        JSR     PUT
                        ; GET NEXT CHARACTER
                        JMP     SCANS

                        ; CREATE A ZERO
                ISTAR:  SUB     0,0
                        ; INITIALIZE VALUE TO ZERO
                        STA 0   VALUE

                        ; READ A CHARACTER
                SCAND:  JSR     GET
                        ; IS IT A "CR" OR "LF" ?
                        JSR     TCRLF
                        ; YES, VALUE IS COMPLETE
                        JMP     GOTIT
                        ; ASCII "ZERO"
                        LDA 1   ..0
                        ; ASCII "NINE"
                        LDA 3   ..9
                        ; > 9?
                        SGT     0,3
                        ; < 0?
                        SGE     0,1
                        ; NOT A DIGIT, TRY NEXT CHARACTER
                        JMP     SCAND

                        ; CONVERT DIGIT TO BINARY
                        SUB     1,0
                        ; GET OLD VALUE
                        LDA 1   VALUE
                        ; TIMES TWO
                        MOVZL   1,3
                        ; TIMES FOUR
                        MOVZL   3,3
                        ; TIMES TEN
                        ADDZL   3,1
                        ; ADD IN NEW DIGIT
                        ADD     0,1
                        ; UPDATE VALUE
                        STA 1   VALUE
                        ; GET NEXT CHARACTER
                        JMP     SCAND

                        ; PICK UP THIS VALUE
                GOTIT:  LDA 0   VALUE
                        ; AND TARGET VALUE
                        LDA 1   TVAL,2
                        ; ARE THEY THE SAME?
                        SNE     0,1
                        ; NO, TRY NEXT ENTRY
                        JMP     START

                        ; DISK = 1
                        MOVZL   0,0
                        ; RELEASE IT
                        SKP
                        ; GET STRING COUNTER
                FILL:   LDA 1   SCTR
                        ; STRING FULL?
                        MOV     1,1 SNR
                        ; YES, ALL DONE
                        JMP     @TRN,2
                        ; ASCII "SPACE"
                        LDA 0   .40
                        ; FILL OUT STRING WITH SPACES
                        JSR     PUT
                        ; TEST IF FULL YET
                        JMP     FILL

                        ; RECOVER TIT
                ERROR:  LDA 2   2
                        ; POINT TO BEGINNING OF STRING
                        JSR     SETUP
                        ; FILL ENTIRE STRING WITH SPACES & GO AU
                        JMP     FILL
```



```
                        ; READ IN A CHARACTER

                GET:    STA 3   SRET    ; SAVE RETURN ADDRESS
                        LDA 2   BUFF    ; GET BUFFER ADDRESS
                        LDA 3   DDOS    ; SYSTEM TABLE ADDRESS
                        JSR     @D.IN,3 ; READ A CHARACTER
                        JMP     ERROR   ; QUIT ON END-OF-FILE
                        LDA 1   .177    ; 7-BIT MASK
                        AND     1,0     ; EXTRACT DATA
                        LDA 2   2       ; RECOVER TIT
                        JMP     @SRET   ; AND RETURN

                        ; TEST FOR "CR" OR "LF"

                TCRLF:  LDA 1   .CR     ; ASCII "CR"
                        SNE     0,1     ; IS CHARACTER A "CR" ?
                        JMP     0,3     ; YES, RETURN DIRECTLY
                        LDA 1   .LF     ; ASCII "LF"
                        SNE     0,1     ; IS CHARACTER A "LF" ?
                        JMP     0,3     ; YES, RETURN DIRECTLY
                        JMP     1,3     ; NEITHER, SO SKIP-RETURN

                        ; INITIALIZE STRING POINTER & COUNTER

                SETUP:  LDA 1   .AN     ; STRING OFFSET
                        ADDZL   2,1     ; CREATE BYTE POINTER TO STRI
                        STA 1   SPTR    ; INITIALIZE STRING POINTER
                        LDA 1   .24.    ; STRING LENGTH IS 24 CHARACT
                        STA 1   SCTR    ; INITIALIZE STRING COUNTER
                        JMP     0,3     ; RETURN

                        ; INSERT A CHARACTER INTO THE STRING

                PUT:    LDA 1   SCTR    ; PICK UP STRING COUNTER
                        MOV     1,1 SZR ; IS STRING FULL?
                        JMP     0,3     ; YES, IGNORE CHARACTER
                        STA 3   SRET    ; SAVE RETURN ADDRESS
                        LDA 3   SPTR    ; PICK UP STRING POINTER
                        DSZ     SCTR    ; DECREMENT STRING COUNTER
                        ISZ     SPTR    ; INCREMENT STRING POINTER

                        LDA 1   .377    ; HALF-WORD MASK
                        MOVZR   3,3 SNC ; MASK HIGH BYTE
                        SWVS    0,0 SKP ; SWAP DATA OR MASK
                        COM     1,1
                        LDA 2   0,3     ; PICK UP DATA WORD
                        AND     1,2     ; EXTRACT HALF
                        ADD     0,2     ; ADD NEW BYTE
                        STA 2   0,3     ; REPLACE DATA

                        LDA 2   2       ; RECOVER TIT
                        JMP     @SRET   ; AND RETURN
```

AREA NAME LOOKUP (BEAT.)

The area name lookup routine receives as input a beat number in AC0. It searches through the file "XX.PATROL.AREAS" looking for an entry with a matching beat number. If one is found, the corresponding ASCII string is read from the file and stored in the string area of the terminal information table.

If the center name is invalid, the disk file not found, or no matching heat entry found, the string area is set to all blanks.


```
                        .HEAD   A I D S     A R E A     N A M E     L O O K U P
                        .TITLE  BEAT.
                000130  .CCN=   130     ; LOCATE CENTER NAME
                000020  CCN=    20      ; CONTROL CENTER NAME
                000051  RTRN=   R1      ; RETURN ADDRESS
                000062  TGT=    S2      ; TARGET VALUE
                000124  AN=     124     ; STRING OFFSET
                        .ENT    BEAT.
                        .NREL
000000'055051   BEAT.:  STA 3   RTRN,2  ; SAVE RETURN
000001'041062           STA 0   TGT,2   ; AND TARGET
000002'132521           SUBZL   0,0     ; DISK = 1
000003'004050           ENQ             ; RESERVE IT
                        ; VERIFY, LOCATE FILE
000004'021020           LDA 0   CCN,2   ; CONTROL CENTER
000005'006130           JSR     @.CCN   ; LOCATE IN TABLE
000006'000470           JMP     ERROR   ; NOT IN TABLE
000007'025401           LDA 1   1,3     ; DEPARTMENT NAME
000010'030556           LDA 2   FILE    ; FILE NAME POINTER
000011'045000           STA 1   0,2     ; CONSTRUCT FILE NAME
000012'034041           LDA 3   DDOS    ; SYSTEM TABLE ADDRESS
000013'007402           JSR     @D.SRC,3; [illegible]
000014'000462           JMP     ERROR   ; [illegible]
                        ; SETUP FILE FOR INPUT
000015'021010           LDA 0   F.ID,2  ; FILE IDENTIFIER
000016'041454           STA 0   D.SPB,3 ; SAVE FOR PRIM
000017'021013           LDA 0   F.FST,2 ; HIGH-ORDER ADDRESS
000020'025015           LDA 1   F.BFP,2 ; LOW-ORDER ADDRESS
000021'030544           LDA 2   .377    ; HALF-WORD MASK
000022'101220           MOVZR   0,0     ; SHIFT HI BYTE TO LOW, THEN
000023'147500           ANDL    2,1     ; LOW-ORDER, WITH CARRY
000024'030553           LDA 2   BUFF    ; BUFFER ADDRESS
000025'006753           JSR     D.PRM,3 ; WRITE BUFFER POINTER
000026'030002           LDA 2   2       ; RECOVER FIT
```

```
                ; TYPE ERROR MESSAGE

                ERROR:  LDA 1   ERR     ; TYPE "?"
                        LDA 0   .130
                        LDA 3   PT.2    ; RETURN ADDRESS
                        JMP     @ZAP,2  ; EXIT THRU ZAP

                .30:    30
                .130:   130
                ERR:    7*4000+"?

                        .END    OTHER
```


SYMBOL	VALUE	DEF'N	REFERENCES			
CON	000020	1:14	1:29	1:55		
CODE	000040	1:18	1:34			
[illegible]	000041'	2:11	2:04			
ERROR	000033'	2:04	1:32			
JAM	000017'	1:46	1:51			
OTHER	000004'	1:28	2:13			
POST	040041	1:19	1:40			
PT	[illegible]52	1:22	1:29	1:57	2:06	
TIME	[illegible]22	1:15	1:36			
TLOG	000042	1:20	1:42			
[illegible]IT	[illegible]23	1:16	1:30	1:56		
[illegible]	[illegible]24	1:24	1:47			
[illegible]	[illegible]6	1:13	2:07			
.130	[illegible]4'	2:14	2:05			
.30	[illegible]37'	2:09	1:44			
.LOG	[illegible]37	1:09	1:53			
.MSG	[illegible]20	1:07	1:58			
.MSG	[illegible]25	1:[illegible]	1:31			

FLAGGED LINES: NONE

TEN CODE UTILITY (OTHER)

The TEN CODE utility will update the code and time information in the unit status table entry for this unit. The unit must be on duty. Exit is made through the unit status output routine which displays the new status for this unit. This utility is initiated by a call to the invoke function from the unit function.


```
                        .HEAD   A I D S    T E N - C O D E    U T I L I T Y
                        .TITLE  OTHER
                        ;       AIDS TABLE ASSIGNMENTS
000120                  .US0=   120     ; UNIT STATUS OUTPUT
000126                  .US3=   126     ; UNIT STATUS LOCATE
000137                  .LOG=   137     ; MAKE LOG ENTRY
                        ;       TIT WORD ASSIGNMENTS
000006                  ZAP=    6       ; DISPLAY OUTPUT ENTRY
000020                  CCN=    20      ; CONTROL CENTER NAME
000022                  TIME=   22      ; CURRENT TIME OF DAY
000023                  UNIT=   23      ; CURRENT UNIT NUMBER
000040                  CODE=   P4      ; STATUS CODE
000041                  POST=   P1      ; POST NUMBER
000042                  TLOG=   P2      ; TICKET NUMBER
000052                  RT=     R2      ; RETURN ADDRESS
000124                  UT=     124     ; STATUS INFO AREA
                        .NOCL
000004'055052   OTHER:  STA 3   RT,2    ; SAVE RETURN
000001'021020           LDA 0   CCN,2   ; CONTROL CENTER
000002'025023           LDA 1   UNIT,2  ; UNIT NUMBER
000003'006126           JSR     @.US3   ; IF NOT FOUND,
000004'000427           JMP     ERROR   ; COMPLAIN
000005'021040           LDA 0   CODE,2  ; GET THE STATUS
000006'041402           STA 0   2,3     ; INTO THE TABLE
000007'021022           LDA 0   TIME,2  ; CURRENT TIME
000010'041403           STA 0   3,3     ; INTO THE TABLE
000011'025404           LDA 1   4,3     ; POST NUMBER
000012'045041           STA 1   POST,2  ; SET FOR LOG
000013'025405           LDA 1   5,3     ; TICKET NUMBER
000014'045042           STA 1   TLOG,2  ; SET FOR LOG
000015'024422           LDA 1   .30     ; STRING LENGTH
000016'124400           NEGOR   1,1     ; MAKE WORD COUNT
000017'021424   JAM:    LDA 0   24,3    ; OBTAIN A WORD
000020'041124           STA 0   UT,2    ; AND INSERT IT
000021'175400           INC     3,3     ; BUMP ADDRESS
000022'151400           INC     2,2     ; THIS ONE TOO
000023'125404           INC     1,1 SZR ; BUMP COUNTER
000024'000773           JMP     JAM     ; AND REPEAT
000025'031000'000002    LDA 2   2       ; RECOVER TIT
000026'006137           JSR     @.LOG   ; MAKE LOG ENTRY
000027'021020           LDA 0   CCN,2   ; CONTROL CENTER
000030'025023           LDA 1   UNIT,2  ; UNIT NUMBER
000031'035052           LDA 3   RT,2    ; RETURN ADDRESS
000032'000120           JMP     @.US0   ; EXIT THRU STATUS
```

```
                        INC     3,3     ; BUMP ADDRESS
                        INC     2,2     ; THIS ONE TOO
                        INC     1,1 SZR ; BUMP COUNTER
                        JMP     JAM     ; AND REPEAT

                        LDA 2   2       ; RECOVER TIT
                        JSR     @.LOG   ; MAKE LOG ENTRY
                        LDA Ø   CON,2   ; CONTROL CENTER
                        LDA 1   UNIT,2  ; UNIT NUMBER
                        LDA 3   RT,2    ; RETURN ADDRESS
                        JMP     @.USØ   ; EXIT THRU STATUS

                ; TYPE ERROR MESSAGE

                ERROR:  LDA 1   ERR     ; TYPE "?"
                        LDA Ø   .13Ø
                        LDA 3   RT,2    ; RETURN ADDRESS
                        JMP     @ZAP,2  ; EXIT THRU ZAP

                .3Ø:    3Ø
                .13Ø:   13Ø
                ERR:    7*4ØØ+"?

                .EXTN   BEAT.
                .BEAT:  BEAT.

                .END    TEN8
```


SYMBOL	VALUE	DEF'N	REFERENCES			
BEAT.	'X	2:24	2:25			
CON	000020	1:14	1:29	1:36	2:08	
CODE	000040	1:18	1:41			
ERR	000052'	2:22	2:15			
ERROR	000044'	2:15	1:32	1:39		
JAM	000038'	1:57	2:04			
JSR	000041	1:19	1:47			
LDA	000052	1:22	1:28	2:10	2:17	
TEN8	000000'	1:28	2:27			
TIME	000022	1:15	1:43			
TLOG	000042	1:20	1:49			
UNIT	000023	1:16	1:30	1:37	2:09	
UT	000124	1:24	1:57			
ZAP	000006	1:13	2:13			
.13Ø	000051'	2:21	2:16			
.3Ø	000050'	2:20	1:55			
.BEAT	000053'	2:25	1:31			
.LOG	000137	1:09	2:07			
.USØ	000120	1:07	2:11			
.US3	000126	1:08	1:31	1:38		

FLAGGED LINES: NONE

TEN-8 UTILITY (TEN8)

The TEN-8 utility will clear the ticket information, set the code to 14-13 and update the time in the unit status table entry for this unit. The unit must be on duty. Exit is made through the unit status output routine which displays the new status for this unit. This utility is initiated by a call to the invoke function from the unit function.

```
PAGE  1          A I D S     T E N - 8     U T I L I T Y
02/20/74
  1                        .HEAD   A I D S     T E N - 8     U T I L I T Y
  2
  3                        .TITLE  TEN8
  4
  5                        ;       AIDS TABLE ASSIGNMENTS
  6
  7          000120        .US0=   120     ; UNIT STATUS OUTPUT
  8          000126        .US3=   126     ; UNIT STATUS LOCATE
  9          000137        .LOG=   137     ; MAKE LOG ENTRY
 10
 11                        ;       TIT WORD ASSIGNMENTS
 12
 13          000006        ZAP=    6       ; DISPLAY OUTPUT ENTRY
 14          000020        CCN=    20      ; CONTROL CENTER NAME
 15          000022        TIME=   22      ; CURRENT TIME OF DAY
 16          000023        UNIT=   23      ; CURRENT UNIT NUMBER
 17
 18          000040        CODE=   P0      ; STATUS CODE
 19          000041        POST=   P1      ; POST NUMBER
 20          000042        TLOG=   P2      ; TICKET NUMBER
 21
 22          000052        RT=     R2      ; RETURN ADDRESS
 23
 24          000124        UT=     124     ; STATUS INFO AREA
 25
 26                        .NREL
 27
 28  000000'055052 TEN8:   STA 3   RT,2    ; SAVE RETURN
 29  000001'021024         LDA 0   CCN,2    ; CONTROL CENTER
 30  000002'025023         LDA 1   UNIT,2   ; UNIT
 31  000003'006126         JSR     @.US3    ; IN
 32  000004'000440         JMP     ERROR    ;
 33  000005'021406         LDA 0   6,3      ;
 34  000006'006445         JSR     @.BEAT   ; LOOK IF
 35
 36  000007'021024         LDA 0   CCN,2    ; CONTROL
 37  000010'025023         LDA 1   UNIT,2   ; UNIT
 38  000011'006126         JSR     @.US3    ; LOCATE I
 39  000012'000432         JMP     ERROR    ; WILL NOT
 40
 41  000013'021040         LDA 0   CODE,2   ; GET THE STATUS
 42  000014'041402         STA 0   2,3      ; INTO THE TABLE
 43  000015'021022         LDA 0   TIME,2   ; CURRENT TIME
 44  000016'041403         STA 0   3,3      ;
 45
 46  000017'025404         LDA 1   4,3      ; POST
 47  000020'045041         STA 1   POST,2   ; SET FOR
 48  000021'025405         LDA 1   5,3      ; TICKET
 49  000022'045042         STA 1   TLOG,2   ; SET FOR LOG
 50
 51  000023'124444         SUB     0,0
 52  000024'041404         STA 0   4,3      ; CLEAR POST
 53  000025'041405         STA 0   5,3      ; AND TICKET
 54
 55  000026'024422         LDA 1   .30      ; STRING LENGTH
 56  000027'124640         NEGOR   1,1      ; MAKE NEG COUNT
 57  000030'021124 JAM:    LDA 0   UT,2     ;
 58  000031'041424         STA 0   24,3     ; AND
```

VAR	DEF'N	REFERENCES			
	1:[illegible]	2:18			
'X	5:25	5:29			
	1:32	2:14	3:03	4:36	5:19
12'	5:16	4:39			
2'	5:15	5:38			
11'	5:[illegible]	2:37	3:06		
[illegible]	1:3[illegible]	3:43			
[illegible]	1:4[illegible]	4:05			
[illegible]	1:[illegible]	4:[illegible]			
[illegible]	1:12	3:38	4:03		
[illegible]	1:21	4:[illegible]			
[illegible]'	5:35	2:14			
[illegible]'	3:22	3:27			
[illegible]	1:32	4:12	4:14	4:18	
1	1:27	3:[illegible]			
'X	5:[illegible]	5:28			
32	1:[illegible]	2:03	5:10	5:21	
'X	5:[illegible]	5:[illegible]7			
125'	5:1[illegible]	3:33			
[illegible]'	2:[illegible]	4:37			
15'	5:[illegible]	3:13			
22	1:[illegible]	3:15			
'X	[illegible]	5:[illegible]			
24	1:[illegible]	4:24			
42	1:2[illegible]	3:11	3:31		
76'	4:2[illegible]	4:31			
[illegible]	1:07	2:13			
21	1:34	4:14	4:13	4:16	5:03
23	1:2[illegible]	2:[illegible]	3:04	5:20	
[illegible]2	1:1[illegible]	4:[illegible]			
[illegible]3	1:1[illegible]	4:3[illegible]			
24	1:3[illegible]	3:[illegible]			
16	1:1[illegible]	5:11			
123'	5:14	5:[illegible]			
113'	1:[illegible]	3:[illegible]			
112'	5:[illegible]	2:15			
134'	5:2[illegible]	2:[illegible]			
137	1:1[illegible]	3:[illegible]			
133'	5:2[illegible]	5:13			
132'	5:27	3:37			
131'	5:24	4:31			
135'	[illegible]	3:11			
125	1:1[illegible]	4:37			
114'	[illegible]	2:16			
123	1:[illegible]	5:22			
[illegible]	1:11	2:16	3:05		

[illegible] LINKS: NONE

```
                ; DISPOSITION CODE

                        LDA 3   FORM,2  ; TICKET FORMAT
                        LDA 3   T.SA,3  ; FORMAT START
                        LDA 1   DSC,3   ; DISPOSITION FIELD
                        STA 1   FIELD,2
                        SUB     1,1     ; POSITION ZERO
                        STA 1   INDEX,2

                        LDA 1   DP,2    ; DISPOSITION
                        MOVS    1,1
                        JSR     PUT,2   ; FIRST CHARACTER
                        LDA 1   0,2
                        JSR     PUT,2   ; SECOND CHARACTER

                        LDA 1   2+1,2   ; SECOND HALF
                        MOVS    1,1
                        JSR     PUT,2   ; THIRD CHARACTER
                        LDA 1   3+1,2
                        JSR     PUT,2   ; FOURTH CHARACTER

                ; REMOVE TICKET STATUS

                        LDA 1   TKT,2   ; TICKET NUMBER
                        LDA 3   USTB    ; UNIT STATUS TOP
                        LDA 2   0,3     ; TICKET NUMBER
                        SUB#    1,1 SNR ; IF EQUAL,
                        STA 0   0,3     ; CLEAR IT

                        LDA 1   .SIZE   ; RECORD LENGTH
                        ADD     2,3     ; MOVE TO NEXT
                        LDA 0   USIZE   ; END OF TABLE
                        SUB#    2,3     ; IF MORE,
                        JMP     .-5     ; REPEAT

                        LDA 0   CCN,2   ; CONTROL CENTER
                        JSR     @.DS2   ; DELETE TICKET
                        TOP             ; IF NONE, OKAY
                        JMP     CLOUT   ; RETURN
```



```
                        .4:     -4
                        .30:    30
                        .UD:    UD
                        TEN.8:  1000.

                ; TYPE ERROR, RETURN

                ERROR:  LDA 1   ERR     ; TYPE "?"
                        LDA 0   .130
                        LDA 3   RT,2    ; RETURN ADDRESS
                        JMP     @ZAP,2  ; EXIT THRU ZAP

                ERR:    7*400+"?
                .130:   130

                ; PRINT STATUS, RETURN

                CLOUT:  JSR     @.RET   ; CLEAR TICKET
                STOUT:  LDA 0   CCN,2   ; CONTROL CENTER
                        LDA 1   UNIT,2  ; UNIT NUMBER
                        LDA 3   RT,2    ; RETURN ADDRESS
                        JMP     @.DS2   ; EXIT THRU STATUS

                .SIZE:  40      ; STATUS TABLE ENTRY SIZE

                        .EXTN   SET.,RET.,BEAT.,TIME.
                .SET:   SET.
                .RET:   RET.
                .BEAT:  BEAT.
                .TIME:  TIME.

                .TXTM   1
                .TXTX   1

                MSG:    .TXT  *  DISP*

                        .END    TEN24
```

```
                ; ASSIGNMENT COMPLETED

                TEN24:  STA 3   RT,2    ; SAVE RETURN
                        LDA Ø   CCN,2   ; CONTROL CENTER
                        LDA 1   UNIT,2  ; UNIT NUMBER
                        JSR     @.US3   ; LOCATE IN TABLE
                        JMP     ERROR   ; NOT IN TABLE
                        LDA Ø   6,3     ; PATROL AREA
                        JSR     @.BEAT  ; LOOK IT UP

                ; GET DISPOSITION

                        JMP     *TYPE   : TYPE "DISP"
                        MSG
                        LDA Ø   .4
                        LDA 1   .DD     ; DATA OFFSET
                        ADD     2,1     ; MAKE ADDRESS
                        JSR     @ASK    ; GET DISPOSITION
                        NOP             ; IF NONE, OKAY
```



```
                ; UPDATE UNIT STATUS

00016'021020            LDA Ø   CCN,2   ; CONTROL CENTER
00017'025023            LDA 1   UNIT,2  ; UNIT NUMBER
00020'006126            JSR     @.US3   ; LOCATE IN TABLE
00021'000475            JMP     ERROR   ; NOT IN TABLE

00022'025404            LDA 1   4,3     ; POST NUMBER
00023'045041            STA 1   POST,2  ; SET FOR LOG
00024'025405            LDA 1   5,3     ; TICKET NUMBER
00025'045042            STA 1   TLOG,2  ; SET FOR LOG

00026'021467            LDA Ø   TEN.8   ; STATUS IS 10-8
00027'041402            STA Ø   2,3     ; INTO THE TABLE
00030'021022            LDA Ø   TIME,2  ; CURRENT TIME
00031'041403            STA Ø   3,3     ; INTO THE TABLE
00032'122441            SUB     Ø,Ø
00033'041404            STA Ø   4,3     ; CLEAR POST

00034'024457            LDA 1   .30     ; STRING LENGTH
00035'124443            NEGOR   1,1     ; MAKE WORD COUNT
00036'021124    OL:     LDA Ø   UT,2    ; OBTAIN A WORD
00037'041424            STA Ø   24,3    ; AND INSERT IT
00040'175400            INC     3,3     ; BUMP ADDRESS
00041'151401            INC     2,2     ; THIS ONE TOO
00042'125404            INC     1,1 SZR ; BUMP COUNT
00043'000773            JMP     OL      ; AND REPEAT

00044'030002            LDA 2   2       ; RECOVER TIT
00045'006137            JSR     @.LOG   ; MAKE LOG ENTRY
00046'025042            LDA 1   TLOG,2  ; RECOVER TICKET
00047'125015            MOV#    1,1 SNR ; IF NO TICKET,
00050'000455            JMP     STOUT   ; RETURN

                ; COMPLETION TIME

00051'006461            JSR     @.SET   ; SETUP DATA BLOCK
00052'035014            LDA 3   FORM,2  ; TICKET FORMAT
00053'035407            LDA 3   T.SA,3  ; FORMAT START
00054'025771            LDA 1   F24,3   ; COMPLETED FIELD
00055'006464            JSR     @.TIME  ; INSERT TIME
```

TEN-24 UTILITY (TEN24)

The TEN-24 utility is used to indicate that a unit has completed an assignment. The unit must be assigned to a ticket. The message "DISP" is displayed and the operator can type in a disposition code which is written onto the ticket, along with the time of completion. All units assigned to the ticket are released from the ticket and the ticket is taken out of the ticket status table. The unit status table entry is updated for all affected units and the displayed ticket is updated to show disposition and completion time. Exit is made through the unit status output routine which displays the new status for this unit. This utility is initiated by a call to the invoke function from the unit function.


```
                .HEAD   A I D S    T E N - 2 4    U T I L I T Y

                .TITLE  TEN24

                ;       AIDS TABLE ASSIGNMENTS

000104  TYPE=    104     ; TEXT STRING TYPER
000105  ASK=     105     ; DATA STRING INPUT
000120  .US2=    120     ; UNIT STATUS OUTPUT
000125  .TS2=    125     ; TICKET STATUS DELETE
000126  .US3=    126     ; UNIT STATUS LOCATE
000137  .LOG=    137     ; MAKE LOG ENTRY
000152  USTR=    152     ; UNIT STATUS TABLE TOP
000153  USTP=    153     ; UNIT STATUS TABLE END

                ;       TIT WORD ASSIGNMENTS

000006  ZAP=     6       ; DISPLAY OUTPUT ENTRY
000014  FORM=    14      ; TICKET FORMAT BLOCK
000016  FIELD=   16      ; FORMAT FIELD NUMBER
000017  INDEX=   17      ; DATA CHARACTER INDEX
000020  CCN=     20      ; CONTROL CENTER NAME
000022  TIME=    22      ; CURRENT TIME OF DAY
000023  UNIT=    23      ; CURRENT UNIT NUMBER
000024  TKT=     24      ; CURRENT TICKET NUMBER

000041  POST=    P1      ; POST NUMBER
000042  TLOG=    P2      ; TICKET NUMBER
000052  RT=      R2      ; RETURN ADDRESS

000064  OUT=     S4      ; DATA OUTPUT VECTOR

000120  UD=      120     ; DISPOSITION AREA
000124  UT=      124     ; STATUS INFO AREA

                ; FORMAT BLOCK FIELD INDICES

177770  F24=     -10     ; COMPLETED FIELD INDEX
177762  FDC=     -16     ; DISPOSITION FIELD INDEX

                .NREL
```

```
                .30:    30
                .UT:    UT

                ; TYPE ERROR, RETURN

                ERROR:  JSR     @.RET   ; CLEAR TICKET
                        LDA 1   ERR     ; TYPE "?"
                        LDA 1   .130
                        LDA 3   PT,2    ; RETURN ADDRESS
                        JMP     ZAP.2   ; EXIT THRU ZAP

                ERR:    7+400+"?
                .13:    13

                ; PRINT STATUS, RETURN

                CLOUT:  JSR     @.RET   ; CLEAR TICKET
                        LDA 0   CCN,2   ; CONTROL CENTER
                        LDA 1   UNIT,2  ; UNIT NUMBER
                        LDA 3   PT,2    ; RETURN ADDRESS
                        JMP     .USR    ; EXIT THRU STATUS

                .EXIT   SET.,RET.,TIME.
                .SET:   SET.
                .RET:   RET.
                .TIME:  TIME.

                .ZA     TEN23
```


SYMBOL	VALUE	DEF'N	REFERENCES			
CCN	000020	1:18	2:04	3:27	4:18	
CLOUT	000101'	4:17	3:47			
CODE	000040	1:22	3:32			
COUNT	000060	1:28	3:14	3:23		
ERR	000077'	4:12	4:07			
ERROL	000072'	4:06	3:30			
ERROR	000073'	4:07	2:07	2:14		
F23	177771	1:36	2:19			
FIELD	000016	1:16	3:36			
FIL	177775	1:35	3:05			
FOR1	000014	1:15	2:17	3:03		
IL	000033'	3:15	3:24			
INDEX	000017	1:17	3:08			
OL	000035'	3:39	3:44			
POINT	000061	1:29	3:13	3:16	3:22	
POST	000041	1:23	2:10			
PUT.	'X	4:23	4:25			
PT	000052	1:26	2:03	4:09	4:20	
SET.	'X	4:23	4:24			
TEN23	000004'	2:03	4:28			
TIME	000022	1:19	3:34			
TIME.	'X	4:23	4:26			
TLOG	000042	1:24	2:12			
UNIT	000023	1:20	2:05	3:28	4:19	
UT	000124	1:31	3:39	4:02		
ZA	000006	1:14	4:10			
.130	000100'	4:13	4:08			
.30	000074'	4:01	3:09	3:37		
.SET	000106	1:07	3:15			
.LOG	000137	1:10	3:26			
.RET	000107'	4:25	4:06	4:17		
.SET	000106'	4:24	2:16			
.TIME	000113'	4:26	2:22			
.USR	000120	1:08	4:21			
.US3	000126	1:09	2:05	3:29		
.UT	000071'	4:02	3:11			

FLAGGED LINES: NONE

```
                ; ARRIVED ON-SCENE

                TEN23:  STA 3   RT,2    ; SAVE RETURN
                        LDA 0   CCN,2   ; CONTROL CENTER
                        LDA 1   UNIT,2  ; UNIT NUMBER
                        JSR     @.US3   ; LOCATE IN TABLE
                        JMP     ERROR   ; NOT IN TABLE

                        LDA 1   4,3     ; POST NUMBER
                        STA 1   POST,2  ; SET FOR LOG
                        LDA 1   5,3     ; TICKET NUMBER
                        STA 1   TLOG,2  ; SET FOR LOG
                        MOV#    1,1 SNR ; IF ZERO,
                        JMP     ERROR   ; COMPLAIN

                        JSR     @.SET   ; SETUP DATA BLOCK
                        LDA 3   FORM,2  ; TICKET FORMAT
                        LDA 3   T.SA,3  ; FORMAT START
                        LDA 1   F23,3   ; ARRIVED FIELD
                        JSR     @.TIME  ; INSERT TIME
```



```
1                       ; INCIDENT LOCATION
2
3   00020'035014        LDA 3   FORM,2  ; TICKET FORMAT
4   00021'035407        LDA 3   T.SA,3  ; FORMAT START
5   00022'025775        LDA 1   FIL,3   ; LOCATION FIELD
6   00023'045016        STA 1   FIELD,2 ; SET INTO TIT
7   00024'102400        SUB     0,0
8   00025'041017        STA 0   INDEX,2 ; POSITION ZERO
9   00026'020442        LDA 0   .30     ; STRING LENGTH
10  00027'041060        STA 0   COUNT,2 ; SET UP COUNTER
11  00030'024441        LDA 1   .UT     ; STRING ADDRESS
12  00031'147120        ADDZL   2,1     ; BYTE POINTER
13  00032'045061        STA 1   POINT,2 ; SAVE FOR LATER
14
15  00033'006106  IL:   JSR     @.GET   ; GET A CHARACTER
16  00034'035061        LDA 3   POINT,2 ; PICK UP POINTER
17  00035'175220        MOVZR   3,3     ; MAKE ADDRESS
18  00036'021400        LDA 0   0,3     ; GET DATA WORD
19  00037'101302        MOVS    0,0 SZC ; SWAP: IF ZERO,
20  00040'107000        ADD     0,1     ; COMBINE BYTES
21  00041'045400        STA 1   0,3     ; REPLACE DATA
22  00042'011061        ISZ     POINT,2 ; BUMP POINTER
23  00043'015060        DSZ     COUNT,2 ; AND COUNT,
24  00044'000767        JMP     IL      ; AND REPEAT
25
26  00045'006137        JSR     @.LOG   ; MAKE LOG ENTRY
27  00046'021020        LDA 0   CCN,2   ; CONTROL CENTER
28  00047'025023        LDA 1   UNIT,2  ; UNIT NUMBER
29  00050'006126        JSR     @.JSR   ; LOCATE IN TABLE
30  00051'000421        JMP     ERROL   ; CAN'T HAPPEN
31
32  00052'021040        LDA 0   CODE,2  ; STATUS CODE
33  00053'041402        STA 0   2,3     ; INTO THE TABLE
34  00054'021022        LDA 0   TIME,2  ; CURRENT TIME
35  00055'041403        STA 0   3,3     ; INTO THE TABLE
36
37  00056'024412        LDA 1   .30     ; STRING LENGTH
38  00057'124640        NEGOR   1,1     ; MAKE WORD COUNT
39  00060'021124  OL:   LDA 0   UT,2    ; OBTAIN A WORD
40  00061'041424        STA 0   24,3    ; AND INSERT IT
41  00062'175400        INC     3,3     ; BUMP ADDRESS
42  00063'151400        INC     2,2     ; THIS ONE TOO
43  00064'125404        INC     1,1 SZR ; BUMP COUNTER
44  00065'000773        JMP     OL      ; AND REPEAT
45
46  00066'030002        LDA 2   2       ; RECOVER TIT
47  00067'000412        JMP     CLOUT   ; AND RETURN
```

TEN-23 UTILITY (TEN23)

The TEN-23 utility is used to indicate that a unit has arrived at the scene of an incident. The unit must be both on duty and assigned to a ticket. All relevant information in the ticket data block, the ticket status table entry, the unit status table entry and the displayed ticket is updated. The utility exits through the unit status output routine which displays the new status for this unit. This utility is initiated by a call to the invoke function from the unit function.


```
                .HEAD   A I D S     T E N - 2 3     U T I L I T Y
                .TITLE  TEN23
                ;       AIDS TABLE ASSIGNMENTS
000106  .GET=    106     ; INPUT IN FORMAT
000120  .US0=    120     ; UNIT STATUS OUTPUT
000126  .US3=    126     ; UNIT STATUS LOCATE
000137  .LOG=    137     ; MAKE LOG ENTRY
                ;       TIT WORD ASSIGNMENTS
000006  ZAP=     6       ; DISPLAY OUTPUT ENTRY
000014  FORM=    14      ; TICKET FORMAT BLOCK
000016  FIELD=   16      ; FORMAT FIELD TO GET
000017  INDEX=   17      ; DATA CHARACTER INDEX
000024  CCN=     20      ; CONTROL CENTER NAME
000022  TIME=    22      ; CURRENT TIME OF DAY
000023  UNIT=    23      ; CURRENT UNIT NUMBER
000040  CODE=    P0      ; STATUS CODE
000041  POST=    P1      ; POST NUMBER
000042  TLOC=    P2      ; TICKET NUMBER
000052  RT=      R2      ; RETURN ADDRESS
000060  COUNT=   S0      ; DATA BYTE COUNTER
000061  POINT=   S1      ; DATA BYTE COUNTER
000124  UT=      124     ; STATUS INFO AREA
                ; FORMAT BLOCK FIELD INDICES
177775  FIL=     -3      ; INCIDENT LOCATION
177771  F23=     -7      ; TIME ARRIVED
                .NREL
```

```
                ; SETUP TICKET DATA BLOCK

                SET.:   STA 3   RA,2    ; SAVE RETURN
                        LDA Ø   .DATA   ; DISPLAY ENTRY
                        STA Ø   OUT,2   ; JUST IN CASE
                        LDA Ø   TKT,2   ; CURRENT TICKET
                        STA Ø   SAV1,2  ; SAVE FOR LATER
                        LDA Ø   FORM,2  ; FORMAT BLOCK
                        STA Ø   SAV2,2  ; SAVE IT
                        LDA Ø   DATA,2  ; DATA BLOCK
                        STA Ø   SAV3,2  ; SAVE IT, TOO

                        MOV     Ø,3 SNR ; IF NO DATA,
                        JMP     FIND    ; GO FIND IT
                        LDA Ø   BLK,3   ; TICKET BLOCK
                        MOV#    Ø,Ø SZR ; IF NOT ZERO,
                        JMP     FIND    ; RECALL TICKET
                        LDA Ø   REF,3   ; TICKET NUMBER
                        SUB     1,1     ; IF DESIRED ONE,
                        JMP     @RA,2   ; EXIT

                FIND:   LDA Ø   .PUT    ; FORMAT WRITE
                        STA Ø   OUT,2   ; SAVE FOR LATER
                        SUB     Ø,Ø
                        STA Ø   DATA,2  ; CLEAR DATA BLOCK
                        LDA 3   RA,2    ; RETURN ADDRESS
                        JMP     @.FIND  ; GO FIND TICKET

                ; RESTORE PREVIOUS TICKET

                RET.:   STA 3   RA,2    ; SAVE RETURN
                        LDA 3   DATA,2  ; TICKET BLOCK
                        LDA Ø   SAV3,2  ; OLD DATA BLOCK
                        SEQ     Ø,3     ; IF DIFFERENT,
                        DSZ     USE,3   ; REDUCE COUNT
                        NOP             ; IN CASE OF SKIP
                        STA Ø   DATA,2  ; INTO THE TIT

                        LDA Ø   SAV2,2  ; OLD FORMAT BLOCK
                        STA Ø   FORM,2  ; INTO THE TIT
                        LDA Ø   SAV1,2  ; OLD TICKET
                        LDA 1   TKT,2   ; USED TICKET
                        STA Ø   TKT,2   ; REPLACE TICKET
                        SEQ     Ø,1     ; IF DIFFERENT,
                        STA 1   OLD,2   ; SAVE USED TICKET
                        JMP     @RA,2   ; RETURN ADDRESS

                .END
```


SYMBOL	VALUE	DEF'N	REFERENCES				
BLK	000005	1:29	2:15				
DATA	000015	1:14	2:10	2:25	2:32	2:37	
FIND	000021'	2:22	2:14	2:17			
FORM	000014	1:13	2:08	2:40			
MOD	000007	1:27					
OLD	000025	1:16	2:45				
OUT	000064	1:20	2:05	2:23			
RA	000051	1:18	2:03	2:20	2:26	2:31	2:46
REF	000004	1:30	2:18				
RET.	000027'	2:31	1:32				
SAV1	000065	1:21	2:07	2:41			
SAV2	000066	1:22	2:09	2:39			
SAV3	000067	1:23	2:11	2:33			
SET.	000000'	2:03	1:32				
TKT	000024	1:15	2:06	2:42	2:43		
USE	000003	1:28	2:35				
.DATA	000114	1:08	2:04				
.END	000122	1:09	2:47				
.PUT	000107	1:07	2:22				

FLAGGED LINES: NONE

MESSAGE FUNCTION (MEMO.)

The message function calls the invoke function .CALL to initiate the message utility TO.

```
PAGE  1          A I D S     M E S S A G E   F U N C T I O N
02/20/74
                        .HEAD   A I D S     M E S S A G E   F U N C T I O N
                        .TITLE  MEMO.
                        .EXTD   TYPE    ; MESSAGE TYPER
                        .EXTD   .CALL   ; INVOKE ENTRY
                        .EXTD   TAD     ; SCREEN ADDRESS
                        .EXTD   ZAP     ; DISPLAY ENTRY
000050          RC=     R0      ; RETURN ADDRESS
                        .ENT    MEMO.
                        .NREL
00000'055050    MEMO.:  STA 3   RC,2    ; SAVE RETURN
00001'02#013            LDA 0   .100
00002'024003$           LDA 1   TAD     ; POSITION
00003'017004$           JSR     ZAP,2
00004'036001$           JSR     TYPE    ; TYPE "TO"
                        MSG
                        JSR     @.CALL  ; CALL FUNCTION
                        .+1
                        .TXTM   1
                        .TXT    *TO.*
                MSG:
00012'052117    .TXT    *TO*
00013'000000
00014'010100    .100:   10100
                        .END
```

[illegible]	VALUE	DEFN	REFERENCES
[illegible]	[illegible]'	1:10	1:13
[illegible]	[illegible]12'	1:29	1:21
[illegible]	[illegible]'	1:11	1:14
[illegible]	[illegible]%	1:17	1:18
[illegible]	[illegible]$X	1:35	1:24
[illegible]	[illegible]$X	1:19	1:19
[illegible]	[illegible]14'	1:33	1:17
[illegible]	[illegible]$X	1:36	1:22

[illegible] LINES: NONE

MESSAGE UTILITY (TO.)

The message utility is used to send short, one line messages from one terminal to another. It asks where to send the message and then waits for the message to be input. It sends the message to the other terminal along with the identifier of the terminal of origin. The message utility is initiated by a call to the invoke function from the message function.

```
PAGE  1        A I D S    M E S S A G E   U T I L I T Y
12/2?/74
  1                .HEAD  A I D S    M E S S A G E   U T I L I T Y
  2
  3                .TITLE MEMO.
  4
  5          000102 .CVA=   102     ; CONVERT TO ASCII
  6          000103 .CVB=   103     ; CONVERT TO BINARY
  7          000104 TYPE=   104     ; AIDS TEXT TYPER
  8          000105 ASK=    105     ; AIDS DATA INPUT
  9          000130 .CCN=   130     ; CENTER LOOK-UP
 10          000142 MAD=    142     ; MESSAGE ADDRESS
 11
 12          000000 ID=     0       ; TERMINAL NUMBER
 13          000001 FLAG=   1       ; TERMINAL FLAGS
 14          000002 LINK=   2       ; GENERAL LINK WORD
 15          000006 ZAP=    6       ; DISPLAY OUTPUT
 16          000013 HOME=   13      ; REST POSITION
 17          000020 CCN=    20      ; CONTROL CENTER
 18          000021 OPR=    21      ; OPERATOR NUMBER
 19
 20          000043 D5=     P3
 21          000044 D4=     P4
 22          000045 D3=     P5      ; DECIMAL DIGITS
 23          000046 D2=     P6
 24          000047 D1=     P7
 25
 26          000051 RT=     R1      ; RETURN ADDRESS
 27
 28          000120 TO=     120     ; DESTINATION AREA
 29
 30                 .NREL
 31
 32 000000'055051  MEMO.: STA 3   RT,2    ; SAVE RETURN
 33 000001'021020         LDA 0   CCN,2   ; CONTROL CENTER
 34 000002'000130         JSR     @.CCN   ; LOCATE TERMINAL
 35 000003'000423         JMP     NONE    ; NOT IN SYSTEM
 36 000004'025401         LDA 1   1,3     ; DEPARTMENT NAME
 37 000005'044546         STA 1   MESS    ; INSERT IN MESSAGE
 38 000006'024537         LDA 1   BLANK   ; TWO ASCII BLANKS
 39 000007'044545         STA 1   FILL    ; INSERT IN MESSAGE
 40
 41 000010'025021         LDA 1   OPR,2   ; OPERATOR NUMBER
 42 000011'125016         MOVZ    1,1 SNR ; IF ZERO,
 43 000012'000351         JMP     @RT,2   ; IGNORE
 44 000013'000102         JSR     @.CVA   ; MAKE ASCII
 45
 46                 ; SENDER'S NUMBER
 47
 48 000014'021044         LDA 0   D4,2    ; DIGIT 4
 49 000015'105300         MOVS    0,1     ; MOVE TO TOP
 50 000016'021045         LDA 0   D3,2    ; DIGIT 3
 51 000017'107000         ADD     0,1     ; COMBINE
 52 000020'044535         STA 1   OPR1    ; AND SAVE
 53
 54 000021'021046         LDA 0   D2,2    ; DIGIT 2
 55 000022'105300         MOVS    0,1     ; MOVE TO TOP
 56 000023'021047         LDA 0   D1,2    ; DIGIT 1
 57 000024'107001         ADD     0,1     ; COMBINE
 58 000025'044531         STA 1   OPR2    ; AND SAVE
```

```
                        ; GET DESTINATION

                NONE:   LDA 0   .4
                        LDA 1   .TO         ; DATA OFFSET
                        ADD     2,1         ; MAKE ADDRESS
                        JSR     @ASK        ; GET DESTINATION
                        RSP                 ; IF NONE, OKAY

                        LDA 0   TO+1,2
                        STA 0   D1,2        ; DIGIT 1
                        MOVS    *,*
                        STA 0   D2,2        ; AND 2

                        LDA 1   TO,2
                        STA 0   CITY
                        STA 0   D3,2        ; DIGIT 3
                        MOVS    *,*
                        STA 0   D4,2        ; AND 4

                        SUB     *,*
                        STA 0   D5,2        ; NO DIGIT 5
                        STA 0   COUNT       ; CLEAR COUNT
                        JSR     @.CVB       ; MAKE BINARY
                        STA 1   STAR        ; AND SAVE IT

                        ; GET THE MESSAGE TEXT

                        LDA 0   .MAX        ; MAX. LENGTH
                        LDA 1   .TEXT       ; TEXT ADDRESS
                        ADD     0,1         ; GET MESSAGE
                        JMP     @RT,2       ; IF NONE, EXIT

                        LDA 0   ID,2        ; TERMINAL ID
                        DEQ                 ; RELEASE IT

                        ; SEARCH FOR TAKERS

                LOOP:   LDA 2   LINK,2      ; FOLLOW THE LINK
                        LDA 3   2           ; STARTING POINT
                        SNE     2,3         ; IF EQUAL,
                        JMP     DONE        ; ALL DONE

                        LDA 0   STAR        ; DESIRED NUMBER
                        LDA 1   OPR,2       ; THIS OPERATOR
                        MOV#    0,0 SZR     ; IF NOT ZERO,
                        JMP     TEST        ; TEST THESE

                        LDA 0   CITY        ; DESIRED CENTER
                        LDA 1   CCN,2       ; THIS CENTER
                        MOV#    0,0 SNR     ; IF ZERO,
                        JMP     DOIT        ; DUMP IT HERE

                        LDA 3   LMASK       ; LO-BYTE MASK
                        AND#    3,0 SNR     ; IF NO LOW BYTE
                        JMP     CHOP
                        AND#    3,1 SZR     ; TEST ON HIGH BYTE
                        JMP     TEST
```



```
                CHOP:   LDA 3   HMASK       ; HI-BYTE MASK
                        AND     3,0         ; EXTRACT TOP BYTE
                        AND     3,1         ; OF BOTH NAMES

                TEST:   SEQ     0,1         ; IF NOT EQUAL,
                        JMP     LOOP        ; TRY THE NEXT ONE

                DOIT:   LDA 0   FLAG,2      ; TERMINAL FLAGS
                        MOVL#   0,0 SNC     ; IF A PRINTER,
                        JMP     LOOP        ; JUST MOVE ON

                        ; PRINT MESSAGE

                        LDA 0   ID,2        ; TERMINAL ID
                        ENQ                 ; WAIT FOR IT

                        LDA 1   MAD         ; MESSAGE ADDRESS
                        SUB     *,*
                        JSR     @ZAP,2      ; POSITION CURSOR

                        JSR     @TYPE       ; TYPE MESSAGE
                        MESS

                        LDA 1   HOME,2      ; REST POSITION
                        SUB     0,0
                        JSR     @ZAP,2      ; RETURN CURSOR

                        LDA 0   ID,2        ; TERMINAL ID
                        DEQ                 ; GIVE IT UP

                        ISZ     COUNT       ; BUMP THE COUNT
                        JMP     LOOP        ; AND CONTINUE

                        ; CHECK FOR DELIVERY

                DONE:   LDA 0   ID,2        ; TERMINAL ID
                        ENQ                 ; GET IT BACK
                        LDA 0   COUNT       ; DELIVERY COUNT
                        MOV#    0,0 SZR     ; IF NOT ZERO,
                        JMP     @RT,2       ; RETURN

                        ; TYPE ERROR MESSAGE

                ERROR:  LDA 1   ERR         ; TYPE "?"
                        LDA 0   .TO
                        LDA 3   RT,2        ; RETURN ADDRESS
                        JMP     @ZAP,2      ; EXIT THRU ZAP

                .4:     4
                .TO:    TO
                .TO:    130
                .MAX:   -80.
                .TEXT:  TEXT
                .FXCT:  1
                ERR:    7*400+"?
                BLANK:  " *401
                LMASK:  377*400
```

```
//
```

[illegible]	REF#	REFERENCES				
[illegible]	1:[illegible]	2:37	2:31			
[illegible]	3:57	1:38				
[illegible]	1:17	1:33	2:50			
[illegible]	3:12	2:56				
[illegible]	4:[illegible]	2:16	2:49			
[illegible]	3:[illegible]	2:23	3:32	3:39		
[illegible]	1:[illegible]	1:56	2:11			
[illegible]	1:[illegible]	1:54	2:13			
[illegible]	1:[illegible]	1:50	2:17			
[illegible]	1:21	1:48	2:19			
[illegible]	1:[illegible]	2:22				
[illegible]	3:[illegible]	2:52				
[illegible]	3:37	2:12				
[illegible]	3:[illegible]	3:15				
[illegible]	[illegible]					
[illegible]	1:[illegible]	1:39				
[illegible]	1:13	3:39				
[illegible]	3:13	3:32				
[illegible]	1:16	3:25				
[illegible]	1:12	2:34	3:15	3:29	3:37	
[illegible]	1:14	2:39				
[illegible]	4:[illegible]	2:54				
[illegible]	1:39	3:37	3:11	3:33		
[illegible]	1:[illegible]	3:13				
[illegible]	1:32	4:23				
[illegible]	1:[illegible]	1:37	3:23	4:09		
[illegible]	2:[illegible]	1:35				
[illegible]	1:18	1:41	2:45			
[illegible]	4:[illegible]	1:[illegible]				
[illegible]	4:[illegible]	1:[illegible]				
[illegible]	1:26	1:32	1:43	2:32	3:41	3:47
[illegible]	1:[illegible]	2:25	2:44			
[illegible]	4:[illegible]	2:47	2:53			
[illegible]	3:17	3:54				
[illegible]	1:[illegible]	2:[illegible]	2:15	3:51		
[illegible]	1:17	2:42				
[illegible]	1:[illegible]	3:[illegible]	3:27	3:48		
[illegible]	3:32	3:45				
[illegible]	3:[illegible]	2:[illegible]				
[illegible]	1:[illegible]	1:34				
[illegible]	1:[illegible]	1:44				
[illegible]	1:[illegible]	2:[illegible]				
[illegible]	3:[illegible]	2:[illegible]				
[illegible]	3:[illegible]	2:[illegible]				
[illegible]	4:[illegible]	2:[illegible]				

[illegible] LINES: NONE


```
02/20/74
  1    00147'000377  LMASK:  377
  2    00150'000000  COUNT:  0
  3    00151'000000  CITY:   0
  4    00152'000000  STAR:   0
  5    00153'000000  MESS:   0
  6    00154'000000  FILL:   0
  7    00155'000000  OPR1:   0
  8    00156'000000  OPR2:   0
  9           00153'  .LOC    MESS
 10           000001  .TXTM   1
 11           000001  .TXTN   1
 12    00153'047520  .TXT    *OPERATOR: *
 13    00154'042522
 14    00155'000024
 15    00156'047522
 16    00157'000000
 17    00160'000036  TEXT:   .BLK    60./2
 18    00161'003400          7*400
 19
 20           00000'  .END    MEMO.
```

PRINT FUNCTION (COPY.)

The Print Function asks the operator for a file name, and initiates the print utility to print the named file.

An alternate entry point "CALT." may be used if AC1 contains a file name pointer.


```
                        .HEAD   A I D S    P R I N T    F U N C T I O N
                        .TITLE  COPY.
                        ;       AIDS TABLE ASSIGNMENTS
                        .EXTD   TYPE    ; AIDS TEXT TYPER
                        .EXTD   ASK     ; AIDS DATA INPUT
                        .EXTD   TAD     ; TAG-LINE ADDRESS
                        .EXTD   .KILL   ; DELETE CODE BLOCK
                        ;       TIT WORD ASSIGNMENTS
                        .EXTD   ZAP     ; DISPLAY OUTPUT ENTRY
000000          FILE=   P0      ; FILE NAME POINTER
000054          RC=     R0      ; RETURN ADDRESS
000051          RA=     R1      ; ANOTHER RETURN
000100          FN=     100     ; FILE NAME AREA
                        .ENT    COPY..CALT.
                        .NREL
000001 055054   COPY.:  STA 3   RC,2    ; SAVE RETURN
                        JSR     ..KILL  ; DELETE CODE
                        SUB     0,0
                        LDA 1   TAD     ; POSITION
                        JSR     @ZAP,2
                        JSR     @TYPE   ; TYPE "PRINT"
                        "SJ
                        LDA 0   .20     ; STRING LENGTH
                        LDA 1   .FN     ; STRING OFFSET
                        ADD     2,1     ; MAKE ADDRESS
                        JSR     @ASK    ; GET FILE NAME
                        JMP     @RC,2   ; NO NAME, RETURN
                        LDA 1   .FN     ; STRING OFFSET
                        ADD     2,1     ; MAKE ADDRESS
                        LDA 3   RC,2    ; RETURN ADDRESS
```

```
                        ; TO "AIDS" SYSTEM

000017'035051  CALT.:   STA 3   RA.2    ; SAVE RETURN
000040'045040          STA 1   FILE,2  ; AND FILE NAME
000041'000402          SKIP            ; SKIP AROUND

                        ; INITIATE A PROCESS

000042'126040  TRY:     IDLE            ; WAIT AWHILE
000043'102411          LDA 2   .IOC    ; IOCB ADDRESS
000044'063077          .IO             ; START A PROCESS
000045'000775          JMP     TRY     ; RETRY ON FAILURE
000046'102002          LDA 2   2       ; RECOVER TIT
000047'103411          LDA 3   PCB     ; NEW PCB ADDRESS
000050'101401          MOV 2   PLOC,3  ; SET LOCAL BLOCK
000051'000003          JMP     @RA,2   ; AND RETURN

000012          PLOC=   12      ; PROCESS LOCAL

000020'         .20:    20
000021'         .FN:    FN

000022'000036'  .IOC:   .+1
000023'000000          0
000024'000006          6
000025'000000  PCB:     0
000026'000045'         NAME
000027'000000          0

000030'000001  .TXT     1

000031'         [illegible]:    .TXT    #PRINT#

000035'         NAME:   .TXT    *AIDS.COPY*

                .END
```



```
SYMBOL  VALUE    DEFN    REFERENCES

ASK             $X      1:08    1:37
CALT.   000017'  2:03    1:23
[illegible]     000031'  1:26    1:23
FILE    000040   1:16    2:04
FN      000100   1:21    2:21
MSG     000042'  2:32    1:32
NAME    000045'  2:35    2:27
PCB     000037'  2:26    2:13
PLOC    000012   2:18    2:17
RA      000051   1:19    2:03    2:16
RC      000050   1:18    1:26    1:38    1:41
TAD             $X      1:09    1:22
TRY     000022'  2:09    2:12
TYPE            $X      1:07    1:31
ZAP             $X      1:14    1:33
.20     000032'  2:20    1:31
.FN     000033'  2:21    1:35    1:39
.IOC    000034'  2:23    2:11
.KILL           $X      1:10    1:27

        FLAGGED LINES:  NONE
```

PRINT UTILITY (COPY.)

The print utility is initiated by the print function and copies the input file into the printer file, "XX.PRINTER". On end of file, it writes an end of file and closes the printer file.


```
                        .HEAD   A I D S    P R I N T    U T I L I T Y

                        .TITLE  COPY.

000130          .CCN=   130     ; CONTROL CENTER LOOKUP

000020          CCN=    20      ; CONTROL CENTER NAME
000040          FILE=   P0      ; FILE NAME POINTER

000001          DEPT=   1       ; DEPARTMENT NAME
000010          FLAG=   10      ; PRINTER FILE FLAGS

                        .NOPL

                        ; WAIT FOR PRINTER IDLE

000000'102521   TRY:    SUBZL   0,0     ; DISK = 1
000001'004052           DEQ             ; RELEASE IT
000002'000000           IDLE
000003'102520   COPY.:   SUBZL   0,0     ; DISK = 1
000004'004050           ENQ             ; RESERVE IT

000005'030002           LDA 2   2       ; RECOVER TIT
000006'021020           LDA 0   CCN,2   ; CONTROL CENTER
000007'006130           JSR     @.CCN   ; LOCATE IN TABLE
000010'000455           JMP     DIE     ; NOT IN TABLE

000011'021410           LDA 0   FLAG,3  ; PRINTER FLAGS
000012'101212           MOVR#   0,0 SZC ; IF BUSY,
000013'000765           JMP     TRY     ; WAIT, RETRY
000014'011410           ISZ     FLAG,3  ; SET BUSY

                        ; OPEN PRINTER FILE

000015'025401           LDA 1   DEPT,3  ; DEPARTMENT
000016'030462           LDA 2   NAME    ; FILE NAME
000017'045000           STA 1   0,2     ; SET INTO NAME
000020'024467           LDA 1   BUFF    ; BUFFER ADDRESS
000021'034041           LDA 3   DDOS    ; SYSTEM ADDRESS
000022'007400           JSR     @D.CYS,3; CALL OVERLAY
000023'040006           0.OPN           ; TO OPEN FILE

                        ; SETUP INPUT FILE

000024'030002           LDA 2   2       ; RECOVER TIT
000025'031040           LDA 2   FILE,2  ; FILE NAME
000026'007402           JSR     @D.SRC,3; LOCATE FILE
000027'000422           JMP     EOF     ; NOT FOUND

000030'021010           LDA 0   F.ID,2  ; IDENTIFIER
000031'041454           STA 0   D.SP3,3 ; SAVE FOR PRIME
000032'021013           LDA 0   F.FST,2 ; HIGH-ORDER ADDRESS
000033'025015           LDA 1   F.BFP,2 ; LOW-ORDER ADDRESS
000034'030437           LDA 2   .377    ; LOW-BYTE MASK
000035'101220           MOVZR   0,0     ; SHIFT HIGH-ORDER, TEST
000036'147504           ANDL    2,1     ; LOW-ORDER, WITH MASK
000037'031424           LDA 2   D.BF1,3 ; SYSTEM BUFFER
000040'005753           JSR     D.PRM,3 ; PRIME FOR INPUT
```

```
                ; TRANSFER CONTENTS

          LOOP: LDA 3   DDOS    ; SYSTEM ADDRESS
                LDA 2   D.BF1,3 ; INPUT BUFFER
                JSR     @D.IN,3 ; READ A CHARACTER
                JMP     EOF     ; OOPS, END OF FILE

                LDA 3   DDOS    ; SYSTEM ADDRESS
                LDA 2   BUFF    ; OUTPUT BUFFER
                JSR     @D.OUT,3; WRITE A CHARACTER
                JMP     LOOP    ; AND REPEAT

                ; CLOSE PRINTER FILE

          EOF:  LDA 3   DDOS    ; SYSTEM ADDRESS
                LDA 2   BUFF    ; BUFFER ADDRESS
                JSR     @D.CYS,3; CALL OVERLAY
                .BYTE   D.EOF   ; TO WRITE EOF
                JSR     @D.CYS,3; CALL OVERLAY
                .BYTE   D.CLO   ; TO CLOSE FILE

                ; CLEAN UP, CLEAR OUT

                LDA 2   2       ; RECOVER TIT
                LDA 0   CON,2   ; CONTROL CENTER
                JEQ     *.CON   ; LOCATE IN TABLE
                JMP     DIS     ; NOT IN TABLE
                STA 2   .,0     ; SETS HIGH BIT
                STA 0   FLAG,3  ; CLEAR BUSY, SET FULL

          DIS:  .BYTE   .+1     ; DISK = 1
                0               ; RELEASE IT
                LDA 3   3       ; P2P ADDRESS
                STA 3   P2P     ; SET INTO IOC
                LDA 2   .IOC    ; IOCB ADDRESS
                .IO             ; TERMINATE

                .BYTE   1
                .BYTE   1

          .377: .377

          .+1
                0
                7
          P2P:  0

          NAME: .+1
                .TXT    *XX.PRINTER*

          BUFF: .+1
                .BLK    D.IOA

                .END    COPY.
```


SYMBOL	VALUE	DEF'N	REFERENCES		
BUFF	000107	2:51	1:38	2:09	2:16
CON	000020	1:07	1:24	2:25	
COPY.	000003	1:20	2:54		
DEPT	000001	1:10	1:35		
[illegible]	[illegible]	2:31	1:26	2:27	
[illegible]	[illegible]	2:15	1:33	2:06	
[illegible]	[illegible]	1:08	1:46		
[illegible]	000010	1:11	1:23	1:31	2:29
[illegible]	[illegible]	2:03	2:11		
[illegible]	000100	2:48	1:36		
[illegible]	000077	2:46	2:34		
[illegible]	000100	1:17	1:30		
.377	000073	2:41	1:54		
.CON	000130	1:05	1:25	2:26	
.IOC	[illegible]	2:43	2:35		

FLAGGED LINES: NONE

STATUS FUNCTION (STAT.)

The status function transfers control to the routine whose address is in the end control word of the terminal information table.

```
PAGE  1        A I D S     S T A T U S     F U N C T I O N
02/20/74
                        .HEAD   A I D S     S T A T U S     F U N C T I O N
                        .TITLE  STAT.
                        .EXTD   ZAP     ; DISPLAY COMMAND ENTRY
                        .EXTD   END     ; END CONTROL WORD
                        .ENT    STAT.,END.
                        .NREL
000000'025002$  STAT.:  LDA 1   END,2   ; END WORD
000001'125015           MOV#    1,1 SNR ; IF ZERO,
000002'000403           JMP     .3      ; IGNORE
000003'025002$  END.:   LDA 1   END,2   ; END WORD
000004'127756           ADDO#   1,1 SEZ ; IF VALID,
000005'003002$          JMP     @END,2  ; GO DO IT
000006'024403           LDA 1   .7      ; ASCII BELL-CODE
000007'121401           INC     1,1
000010'003001$          JMP     @ZAP,2  ; RING THE BELL
000011'000007   .7:     7
                        .END
```

[illegible]	[illegible]	REFERENCES			
$X	1:15	1:11	1:15	1:17	
'[illegible]'	1:15	1:[illegible]8			
'[illegible]'	1:11	1:[illegible]8			
$X	1:[illegible]	1:21			
'[illegible]'	1:23	1:19			

FLAGGED LINES: NONE

CLEAR FUNCTION (ZERO.)

The clear function detaches the terminal from whatever tasks it may be attached to and clears the display device.


```
                .HEAD   A I D S    C L E A R    F U N C T I O N
                .TITLE  ZERO.
                .EXTD   .KILL   ; DELETE CODE BLOCK
                .EXTD   ZAP     ; DISPLAY COMMAND ENTRY
                .EXTD   EDIT    ; EDIT TABLE ADDRESS
                .EXTD   DATA    ; DATA BLOCK ADDRESS
                .EXTD   TKT     ; CURRENT TICKET NUMBER
                .EXTD   OLD     ; PREVIOUS TICKET NUMBER
                .EXTD   END     ; END CONTROL WORD
                .EXTD   USE     ; DATA BLOCK USE-COUNT
        000050  RZ=     R0      ; RETURN ADDRESS
                .ENT    ZERO.
                .NREL
000000'055050   ZERO.:  STA 3   RZ,2    ; SAVE RETURN
000001'006001$          JSR     @.KILL  ; DELETE CODE BLOCK
000002'021005$          LDA 0   TKT.2   ; CURRENT TICKET
000003'101014           MOV#    0,0 SZR ; IF NOT ZERO,
000004'041006$          STA 0   OLD.2   ; BECOMES OLD ONE
000005'025004$          LDA 1   DATA.2  ; DATA BLOCK
000006'135004           MOV     1,3 SZR ; IF PRESENT,
000007'015410$          DSZ     USE.3   ; REDUCE USERS
000010'000401           NOP             ; IN CASE OF SKIP
000011'126400           SUB     1,1
000012'045005$          STA 1   TKT.2   ; CLEAR TICKET
000013'045004$          STA 1   DATA.2  ; NO [illegible]
000014'045003$          STA 1   EDIT.2  ; CLEAR [illegible] TABLE
000015'045007$          STA 1   END.2   ; AND [illegible]
000016'020403           LDA 0   ZERO    ; PICK UP FLAGS
000017'035050           LDA 3   RZ,2    ; RESTORE [illegible]
000020'[illegible]      JMP     ZAP,2   ; EXIT [illegible]
000021'140100   ZERO:   140100          ; FLAGS TO CLEAR
                .END
```

```
L  NAME    DEFN   REFERENCES

       $X   1:2     1:27    1:34
       $X   1:6     1:35
       $X   1:12    1:36
       $X   1:11    1:2
     5'     1:16    1:21    1:39
       $X   1:1     1:23    1:3
       $X   1:14    1:29
       $X   1:7     1:1
    2'      1:42    1:38
    1'      1:11    1:1
       $X   1:45    1:22

NUMBER LINES: NONE
```

RADIO FUNCTION (RADIO)

The radio function displays on all terminals (except the terminal initiating the function) a message identifying the initiating terminal and stating whether the radio is wanted or needed. The message is removed when any function key (other than a radio key) is struck on the initiating terminal.


```
                        .HEAD   A I D S     R A D I O     F U N C T I O N
                        .TITLE  RADIO
                        ;       AIDS TABLE ASSIGNMENTS
                        .EXTD   .CVA    ; ASCII CONVERSION
                        .EXTD   TYPE    ; ASCII TEXT OUTPUT
                        .EXTD   .CCN    ; CONTROL CENTER LOOKUP
                        .EXTD   TAD     ; TAG-LINE ADDRESS
                        .EXTD   FAD     ; FLAG-LINE ADDRESS
                        ;       TIT WORD ASSIGNMENTS
                        .EXTD   ID      ; TERMINAL IDENTIFIER
                        .EXTD   FLAG    ; DEVICE FLAGS
                        .EXTD   LINK    ; GENERAL LINK WORD
                        .EXTD   LOOP    ; TIT POINTER
                        .EXTD   EDIT    ; EDIT TABLE ADDRESS
                        .EXTD   SPOT    ; CURRENT CURSOR POSITION
                        .EXTD   HOME    ; CURSOR REST POSITION
                        .EXTD   ZAP     ; DISPLAY OUTPUT ENTRY
                        .EXTD   CCN     ; CONTROL CENTER
                        .EXTD   OPR     ; [illegible]
                        .EXTD   END     ; END CONTROL [illegible]
                P3=     P3
                P4=     P4
                P5=     P5      ; DECIMAL DIGITS
                P6=     P6
                P7=     P7
                R0=     R0      ; RETURN ADDRESS
                S0=     S0      ; RADIO CHANNEL
                S1=     S1      ; PROCESS ADDRESS
                S2=     S2      ; MESSAGE ADDRESS
                I10=    I10     ; STRING OFFSET
                .DIT    ,WANT.,NEED.
                .DWEL
NEED.:  SUB     0,0 SKP ; "NEED RADIO"
WANT.:  ADC     0,0     ; "WANT RADIO"
        STA 0   MA,2    ; MESSAGE FLAG
        STA 3   RC,2    ; SAVE RETURN
        LDA 0   CCN,2   ; CONTROL CENTER
        JSR     @.CCN   ; LOCATE IN TABLE
        JMP     ERROR   ; NOT IN TABLE
        LDA 0   1,3     ; DEPARTMENT NAME
        STA 0   PD,2    ; SAVE FOR LATER
        LDA 0   6,3     ; RADIO CHANNEL
        STA 0   CN,2    ; SAVE THIS ALSO
```

```
                SUB     0,0
                STA 0   EDIT,2  ; CLEAR EDIT TABLE
                LDA 1   TAD     ; POSITION
                JSR     .ZAP,2
                JSR     @TYPE   ; TYPE "RADIO"
                .TXT
                LDA 1   SPOT,2  ; POSITION
                LDA 0   ERASE   ; AND ERASE
                JSR     .ZAP,2

                LDA 1   OPR,2   ; OPERATOR NUMBER
                JSR     .DVA    ; MAKE ASCII

                LDA 1   D4,2    ; DIGIT 4
                MOVS    0,1     ; TO TOP BYTE
                LDA 0   D3,2    ; DIGIT 3
                ADD     0,1     ; COMBINE
                STA 1   N1,2    ; AND SAVE

                LDA 1   D2,2    ; DIGIT 2
                MOVS    0,1     ; TO TOP BYTE
                LDA 0   D1,2    ; DIGIT 1
                ADD     0,1     ; COMBINE
                STA 1   N2,2    ; AND SAVE

                LDA 1   BLANK   ; ASCII BLANKS
                STA 1   P1,2
                STA 1   P2,2    ; PUT IN STRING
                LDA 1   BELL
                STA 1   P3,2    ; BELL AT END

                LDA 1   ID,2    ; TERMINAL ID
                NIO             ; RELEASE OWN
                .TXT
                LDA 1   .7      ; CHANNEL SEVEN
                NIO             ; CONTROLS RADIO

                LDA 0   FLAG,2  ; DEVICE FLAGS
                MOVL#   0,0 SZC ; IF RADIO FLAG,
                JMP     @.+1    ; NEED IN FORCE
                .TXT    NA,2    ; IF "NEED RADIO",
                ISZ     FLAG,2  ; SET RADIO FLAG
```



```
                MOV     2,3 SKP ; TIT ADDRESS
TRY:            LDA 3   LOOP,2  ; TIT POINTER
                LDA 3   LINK,3  ; NEXT TIT
                SNE     2,3     ; IF EQUAL,
                JMP     DONE    ; QUIT
                LDA 0   FLAG,3  ; DEVICE FLAGS
                MOVL#   0,0 SNC ; IF NOT CRT,
                JMP     TRY+1   ; TRY NEXT

                STA 3   LOOP,2  ; TIT POINTER
                LDA 0   CCH,3   ; CONTROL CHARTER
                JSR     .+1CH   ; LOCATE IN TABLE
                JMP     TRY     ; NOT IN TABLE
                LDA 1   6,3     ; RADIO CHANNEL
                LDA 0   CH,2    ; CHANNEL SET
                SEQ     0,1     ; IF NOT EQUAL,
                JMP     TRY     ; TRY NEXT

                LDA 2   LOOP,2  ; REPLACE TIT
                LDA 1   T.2     ; [illegible]
                ...             ; [illegible]
                LDA 1   FCH,2   ; [illegible]
                LDA 3   ADL     ; SET-UP
                SUBZ    0,1     ; = 1
                SNR     1,3     ; IF EQUAL,
                JSR     @SP     ; CLEAR IT OUT
                LDA 1   FCH
                LDA 0   CH      ; [illegible]
                ADD     1,1
                SUB     0,1     ; SET FLAGS
                JSR     @ZAP,2  ; REPOSITION

                LDA 3   2       ; CALLING TIT
                LDA 1   .+2     ; DATA OFFSET
                ADD     3,1     ; TAKE ADDRESS
                STA 1   .+2     ; STORE IN-LINE
                JSR     @TYPE   ; TYPE MESSAGE
                XXX             ; STRING ADDRESS

                LDA 3   2       ; CALLING TIT
                LDA 1   FLAG,3  ; DEVICE FLAGS
                MOVZR   0,0 SNC ; IF RADIO FLAG,
                JMP     WANT    ; FLASH "NEED"
NEED:           JSR     @TYPE   ; "NEED RADIO"
                .TXT
                ISZ     FLAG,2  ; SET RADIO FLAG
                JMP     OKAY
WANT:           JSR     @TYPE   ; "WANT RADIO"
                .TXT

OKAY:           LDA 1   HOME,2  ; MOST POSITIVE
                ...             ; [illegible]
                JSR     @SP,2   ; POSITION
                LDA 1   ID,2    ; TERMINAL ID
                ...             ; [illegible]
                LDA 2   2       ; CALLING TIT
                JMP     TRY     ; TRY NEXT
```

```
                IDLE
DONE:   LDA 3   .GO     ; STARTING ADDRESS
        STA 3   .PCB1   ; INTO IOC BLOCK
        LDA 2   .IOC1   ; IOC BLOCK ADDRESS
        .IO             ; START A PROCESS
        JMP     .-2     ; "CON'T" EXIT

        LDA 2   2       ; RECOVER TIT
        LDA 3   .PCB1   ; PROCESS BLOCK
        STA 2   PAC2,3  ; SET PROCESS AC2
        STA 2   PLOC,3  ; AND LOCAL BLOCK

        LDA 1   ALL     ; END ADDRESS
        STA 1   END,2   ; SET END CONTROL
        STA 3   PA,2    ; SAVE PROCESS
        LDA 0   .7      ; CHANNEL SEVEN
        DEV             ; CONTROLS RADIO
        JMP     0RC,2   ; RETURN

        LDA 2   2       ; RESTORE TIT
ERR1:   LDA 0   .7      ; CHANNEL SEVEN
        .               ; RESTORE TIT

ERROR:  LDA 1   .7      ; ASCII "BELL"
        INC     1,0     ; SET FLAGS
        LDA 3   RC,2    ; GET RETURN
        JMP     @ZAP,2  ; EXIT THRU ZAP

ALL:    .+1             ; END ADDRESS

        SUB     0,0
        STA 0   END,2   ; CLEAR END WORD
        LDA 0   ERASE
        LDA 1   TAD     ; ERASE "RADIO"
        JMP     @ZAP,2
```



```
                PD=     MO      ; DEPARTMENT
                B1=     MO+1    ; TWO BLANKS
                N1=     MO+2    ; OPERATOR...
                N2=     MO+3    ; ...
                B2=     MO+4    ; TWO BLANKS
                B3=     MO+5    ; STRING END

                PAC2=   6       ; PROCESS CONTROL AC2
                PLOC=   12      ; PROCESS LOCAL BLOCK

002000 000070   COL:    80.-24.
002001 020040   BLANK:  "  "
002002 010100   ERASE:  101000
002003 010000   CLEAR:  100000
002004 003400   BELL:   7*400
002005 002242   .GO:    @GO     ; PROCESS START
002006 000110   .MO:    MO      ; STRING OFFSET
002007 000007   .7:     7

                .TXTM   1
                .TXTX   1

                MSG=    .+3
002010 056116   NRM:    .TXT    *\NEEDS\RADIO*
002017 020127   WRM:    .TXT    * WANTS RADIO*

002226 002227   .IOC1:  .+1     ; IOC TO START
002227 000003                   0
002230 000004                   4
                .PCB1:  .
002232 000000                   0
002233 000000                   0

002234 002235   .IOC2:  .+1     ; IOC TO STOP
002235 000000                   0
002236 000007                   7
                .PCB2:  .
```

```
                WAIT:   LDA Ø   .7      ; CHANNEL SEVEN
                        [illegible]     ; CONTROLS RADIO
                CO:     SUB     Ø,Ø     ; CLOCK = Ø
                        END             ; WAIT 1 SECOND
                        SUB     Ø,Ø     ; CLOCK = Ø
                        DEQ             ; [illegible] CLOCK
                        LDA     .7      ; CHANNEL SEVEN
                        END             ; [illegible] RADIO

                        LDA Ø   END,2   ; END CONTROL
                        LDA 1   ALL     ; SET-VALUE
                        [illegible]     ; IF NOT EQUAL,
                        JMP     QUIT    ; THAT'S ALL
                        [illegible] 3   ; CURRENT PROCESS
                        [illegible] 2   ; CONTROLLING ONE
                        DEQ     1,3     ; IF NOT EQUAL,
                        JMP     DIE     ; ABRUPT EXIT

                        LDA Ø   FLAG,2  ; [illegible] FLAGS
                        MOVR#   [illegible] ; IF RADIO FLAG,
                        [illegible]     ; [illegible]
                        [illegible] 2,3 ; [illegible]
                [illegible]:    LDA 3   FLAG,3  ; DEVICE FLAGS
                        MOVL#   1,1,SNC ; IF NOT CRT,
                        JMP     [illegible]+2   ; NEXT TIT

                        LDA 3   LOOP,2  ; SAVE TIT POINTER
                        LDA Ø   [illegible] ; CONTROL CONTROL
                        [illegible]     ; LOOK IT UP
                        [illegible]     ; NOT IN TABLE
                        LDA 1   [illegible] ; RADIO CHANNEL
                        [illegible]     ; [illegible]
                        [illegible]     ; IF NOT EQUAL,
                        [illegible]     ; NEXT TIT

                        [illegible] FLAG,2  ; [illegible] TIT
                        [illegible] ID,2    ; TERMINAL ID
                                        ; WAIT FOR IT
                        LDA 1   .7      ; SEND "CALL"
                        DEQ     1,1     ; SET FLAGS
                        [illegible]     [illegible],2   ; RING THE BELL
                        [illegible]     ID,2    ; TERMINAL ID
                        [illegible]             ; [illegible]

                DONE:   LDA 2   2       ; RESTORE TIT
                        LDA 3   LOOP,2  ; GET TIT POINTER
                        LDA 3   LINK,3  ; MOVE TO NEXT TIT
                        [illegible] 2,3 ; IF NOT FIRST,
                        [illegible]     RING    ; RING ANOTHER
                        [illegible]     WAIT    ; WAIT A SECOND
```



```
                QUIT:   MOVL#   C,C SZC ; IF NEGATIVE,
                        JMP     DIE     ; ABRUPT EXIT
                        MOV     2,3     ; FIRST TIT
                KILL:   LDA Ø   FLAG,3  ; DEVICE FLAGS
                        MOVL#   Ø,Ø SNC ; IF NOT CRT,
                        JMP     [illegible]+2   ; NEXT TIT

                        STA 3   LOOP,2  ; SAVE TIT POINTER
                        LDA Ø   CONT,3  ; [illegible]
                        JSR     [illegible] ; LOOK IT UP
                        JMP     BUMP    ; NOT IN TABLE
                        LDA 1   6,3     ; [illegible]
                        LDA Ø   CH,3    ; GET TO CHA
                        SEQ     C,1     ; IF NOT EQUAL,
                        JMP     [illegible] ; NEXT TIT

                        LDA 2   LOOP,2  ; [illegible] TIT
                        LDA Ø   ID,2    ; [illegible] ID
                        END             ; WAIT FOR IT
                        LDA 1   RED
                        LDA Ø   CRL     ; [illegible]
                        ADD     C,1
                        LDA Ø   CLEAR   ; GET START FLAG
                        JSR     [illegible],2   ; CLEAR IT OUT
                        LDA Ø   FLAG,2  ; DEVICE FLAGS
                        [illegible]     ; IF RADIO FLAG,
                        DSZ     FLAG,2  ; CLEAR IT OUT
                        LDA 1   [illegible],2   ; REST POSITION
                        [illegible]     ,
                        JMP     [illegible],2   ; [illegible]
                        LDA 3   ID,2    ; [illegible]
                        [illegible]             ; CLEAR IT
                BUMP:   LDA 2   2       ; RESTORE TIT
                        LDA 3   LOOP,2  ; GET TIT POINTER
                        LDA 3   LINK,3  ; [illegible]
                        [illegible] 2,3 ; IF NOT FIRST,
                        JMP     KILL    ; CLEAR ANOTHER

                DIE:    LDA 1   .7      ; [illegible]
                        DEQ             ; CLEAR ALL DEVICES
                        LDA 3   3       ; [illegible]
                        STA 3   .POB2   ; SET INTO IOC
                        LDA 2   .IOC2   ; IOC BLOCK ADDRESS
                        .IO             ; [illegible]

                        .END
```

		DEF'N	REFERENCES						
	000172'	4:02	3:23	4:13	6:11				
	000111	5:02	2:27						
	000114	5:05	2:28						
	000204'	5:05	2:29						
	000211'	5:13	2:26						
	000353'	7:02	7:06	7:11	7:15				
	$X	1:20	1:47	3:11	6:28	7:09			
	000203'	5:14	7:23						
	000001	1:03	1:54	3:15	6:32	7:13			
	000217'	5:11	3:23	7:21					
	000047	1:01	2:22						
	000046	1:01	2:20						
	000045	1:00	2:16						
	000044	1:00	2:14						
	000043	1:00							
	000360'	4:01	6:17	7:02					
	000144'	3:04	3:05						
	$X	1:11	2:12						
	$X	1:09	3:22	4:14	4:32	6:10			
	000202'	4:10	2:33	4:33					
	000164'	4:01	2:40						
	000143'	4:23	4:36						
	000165'	4:24	1:49						
	000115	5:06	2:30						
	$X	1:11	3:27	7:20					
	$X	1:16	2:38	2:42	3:06	3:41	3:46	6:19	6:23
			7:04	7:25	7:27				
	000202'	6:03	5:16						
	$X	1:21	3:51	7:28					
	$X	1:15	2:32	3:20	3:54	6:37	6:42	7:18	7:31
	000320'	7:04	7:38						
	$X	1:17	3:03	6:47	7:36				
	$X	1:18	3:02	3:10	3:19	6:27	6:36	6:46	7:08
			7:17	7:35					
	000162	1:36	1:44	2:41					
	000110	1:37	5:01	5:02	5:03	5:04	5:05	5:06	5:17
	000213'	5:23	2:26						
	000112	5:03	2:13						
	000113	5:04	2:24						
	000124'	3:44							
	000040'	1:42	1:39						
	000307'	6:45	6:25	6:30	6:34				
	000217'	5:24	3:45						
	000134'	3:51	3:47						
		1:00	2:11						
		1:35	1:15	6:15					
		1:01	1:42						
		1:02	1:11						
		1:00	2:13						
		1:35	1:36	4:18	4:26				
		1:13	2:12						
		1:00	2:07						
		1:00	2:03	4:34					
			3:	3:13	3:17	3:57			

SYMBOL	VALUE	DEF'N	REFERENCES						
TYPE	$X	1:08	2:05	3:37	3:44	3:48			
.AIT	000240'	6:01	6:21	6:50					
.ANT	000132'	3:48	3:43						
.ANT.	000001'	1:43	1:39						
.ATM	000217'	5:25	3:42						
ZAP	$X	1:22	2:11	2:09	3:31	3:53	4:27	4:35	4:41
			7:23	7:30					
./	000207'	5:18	2:31	4:16	4:21	4:24	6:01	6:07	6:11
			7:41						
.CCN	$X	1:09	1:43	3:12	6:29	7:10			
.CVA	$X	1:07	2:10						
.GO	000205'	5:16	4:00						
.IOC1	000226'	5:27	4:01						
.IOC2	000234'	5:34	7:43						
.JO	000206'	5:17	3:31						
.PC81	000231'	5:30	4:02	4:09					
.PC82	000237'	5:37	7:43						

FLAGGED LINES: NONE

DEBUG FUNCTION (DEBUG)

The debug function loads the debugging routine "AIDS.DEBUG" and transfers control to the debugger. This function is only allowed from the console terminal.


```
                        .HEAD   A I D S     D E B U G     F U N C T I O N
                        .TITLE  DEBUG
                        .EXTD   ID          ; TERMINAL IDENTIFICATION
                        .ENT    TEST.
                        .NREL
000000'000010                   1
000001'020777   TEST.:  LDA 0   .-1         ; CONSOLE ID
000002'025001$          LDA 1   ID,2        ; TERMINAL ID
000003'106513           SUB#    0,1         ; IF NOT EQUAL.
000004'001403           JMP     .+3         ; [illegible]
000005'055050           STA 3   [illegible] ; [illegible]
000006'034434           LDA 3   [illegible] ; [illegible]
000007'175014           MOV#    3,3 SZR     ; IF [illegible]
000010'[illegible]411           JMP     GO          ; GO TO IT
000011'030423           LDA 2   .IOC        ; IOCB ADDRESS
[illegible]             [illegible]         ; LOAD FUNCTION
[illegible]             [illegible]         ; SET INTO IOCB
[illegible]             .IO                 ; [illegible]
[illegible]             JMP     NO          ; WON'T FIT
[illegible]             LDA 2   2           ; RESTORE AC2
[illegible]             LDA 3   .TOB        ; BLOCK ADDRESS
[illegible]             STA 3   TEST        ; SAVE FOR LATER
[illegible]     GO:     JMP     @3          ; [illegible]
[illegible]             JMP     @TEST,2     ; [illegible]
000023'030411           LDA 2   .IOC        ; [illegible]
[illegible]             LDA 1   DEAL        ; [illegible]
[illegible]             SUB 1   1,2         ; [illegible]
[illegible]             .IO                 ; [illegible]
[illegible]             ADDZ#   [illegible] ; CAN'T [illegible]
[illegible]     NO:     LDA 2   2           ; RESTORE AC2
[illegible]             [illegible]         ; [illegible]
[illegible]             STA 3   TEST        ; [illegible]
[illegible]             JMP     @TEST,2     ; GET OUT
[illegible]     .IOC:   .+1                 ; I/O CONTROL BLOCK
[illegible]             [illegible]
[illegible]             [illegible]
000037'000000   .TOB:   0
[illegible]             [illegible]
[illegible]             [illegible]
[illegible]     TEST:   0                   ; DEBUGGER ADDRESS
000043'000006   DEAL:   6                   ; LOAD FUNCTION CODE
[illegible]     [illegible]:    1           ; DEALLOCATE FUNCTION
[illegible]     .[illegible]    1
[illegible]     .[illegible]    1           ; [illegible]
[illegible]     [illegible]:    .TXT        "[illegible]"
                        .END
```

[illegible] A I D s D E F J O F U N C T I O N
[illegible]

[illegible] VALUE XREF%D REFERENCES

[illegible] 1:50 1:31
[illegible] 1:[illegible] 1:19
[illegible] 1:[illegible] 1:12
[illegible] 1:52 1:22
[illegible] 1:[illegible] 1:[illegible]
[illegible] 1:[illegible] 1:25
[illegible] 1:51 1:17 1:29 1:41
[illegible] 1:11 1:17
[illegible] 1:44 1:21 1:33
[illegible] 1:47 1:[illegible]

[illegible] LITES: NONE

LEADS LINE CONTROL (LEADS)

LEADS line control is a loop that monitors the LEADS line. If it receives a "Start of Message" command, it transfers control to the LEADS line receive routine "LRX." If it receives an "Invitation to transmit" command, it transfers control to the LEADS line transmit routine "LTX."


```
                        .HEAD   A I D S     L E A D S   L I N E
                        .TITLE  LEADS
                        .EXTD   ZEP                 ; LINE INPUT CONTROL
                        .EXTN   LRX..LTX.           ; SEND/RECEIVE LOGIC
                        .ENT    LEADS
                        .NREL
                        ; CONTROL-MODE LOOP
00000'007001$ LEADS:    JSR     @ZEP,2              ; WAIT FOR INPUT
00001'020411            LDA 0   .SOM                ; START-OF-MESSAGE
00002'106415            SNE     0,1                 ; IF THAT'S WHAT IT IS,
00003'002405            JMP     @.LRX               ; THEN GO RECEIVE IT
00004'020407            LDA 0   .ITT                ; INVITATION-TO-TRANSMIT
00005'106415            SNE     0,1                 ; IF SO,
00006'002403            JMP     @.LTX               ; THEN GO TRANSMIT
00007'000771            JMP     LOOP                ; IF NEITHER, IGNORE IT
00010'177777 .LRX:      LRX.                        ; LEADS RECEIVE LOGIC
00011'177777 .LTX:      LTX.                        ; LEADS TRANSMIT LOGIC
00012'000001 .SOM:      "A&77                       ; START-OF-MESSAGE CH
00013'000030 .ITT:      "X&77                       ; INVITATION-TO-TRANS
                        .END
```

[illegible] L E A D S L I N E C O N T R O L

[illegible]

[illegible]	[illegible]	[illegible]	[illegible]
[illegible]	1:14	1:22	1:24
[illegible]	1:17	1:25	
[illegible]	1:21	1:27	
[illegible]	1:26	1:31	
[illegible]	1:3[illegible]	1:34	
[illegible]	1:36	1:14	
[illegible]	1:37	1:42	
[illegible]	1:4[illegible]	1:46	

[illegible]

LEADS LINE RECEIVE (LRX.)

The LEADS line receive routine, called by the LEADS line control routine, will accept messages from LEADS bearing the proper drop-code, send acknowledgement, copy the message onto the end of the file "AIDS.LEADS.RX", and return to LEADS line control.


```
                        .HEAD   A I D S    L E A D S    L I N E
                        .TITLE  LRX.
                        .EXTD   CCTB,CCTE           ; CONTROL TABLE
                        .EXTD   LDF                 ; LEADS OUT FILE
                        .EXTD   ZRP,ZDP             ; INPUT/OUTPUT
                        .EXTN   LEADS               ; CONTROL LOOP ENTRY
                        .ENT    LRX.
                        .NREL
00000 000000 000000$ LRX.:  JST     ZRP,2       ; SET UP
00001 000522            LDA 0   .177            ; DATA BYTE MASK
00002 123400            AND     1,0             ; EXTRACT SELECT
00003 034001$           LDA 3   CCTB            ; CONTROL TABLE
00004 025405    LOOK:   LDA 1   DROP,3          ; LOOK
00005 106415            SNE     0,1             ; IF
00006 000000            JMP     0               ; ANOTHER
00007 024513            LDA 1   .40             ; TRY
00010 137000            ADD     1,3             ;
00011 024002$           LDA 1   CCTE            ;
00012 136512            SLE     1,3             ; IF
00013 000771            JMP     .-7             ; TRY
00014 024503            LDA 1   ALL             ; ALL
00015 106414            SEQ     0,1             ;
00016 002507            JMP     @.LDS           ;
00017 000477    OURS:   STA 0   SAVE            ; SAVE
                ; ALLOCATE BUFFER SPACE
00020 000402            SKIP                    ; SKIP AROUND
00021 000000            IOLE                    ; WAIT
00022 030504            LDA 2   .IOCI           ; IOCB ADDRESS
00023 000000            .IO                     ;
00024 000775            JMP     .-3             ; KEEP TRYING
00025 024504            LDA 1   BLK1            ;
00026 044511            STA 1   BLK2            ;
00027 020475            LDA 0   .FWB            ;
00030 107000            ADD     0,1             ;
00031 000464            STA 1   BUFF            ;
                ; OPEN FILE FOR OUTPUT
00032 102520            SUBZL   0,0             ; DISK = 1
00033 004050            ENQ                     ; RESERVE IT
00034 000000$           LDA 1   LDF             ; LEADS OUT FLAG
00035 111201            MOVZL   1,2             ; IF FILE
00036 102561            SUBCL   0,0             ; OPEN AT START
00037 000000            LDA 2   FILE            ; FILE
```

```
                A I D S    L E A D S    L I N E    R E C E I V E
                        LDA 3   DDOS    ; SYSTEM TABLE
                        JSR     @0.CYS,3; CALL OVERLAY
                        0.OPN           ; TO OPEN THE FILE

                ; WRITE SELECT CHARACTER

                        LDA 0   SAVE    ; RETURN SELECT
                        JSR     @D.OUT,3; AND WRITE IT OUT
                        SUBZL   0,0     ; DISK = 1
                        DEO             ; RELEASE IT

                ; ACKNOWLEDGE SELECT

                        LDA 2   2       ; RECOVER TIT
                        LDA 1   ACK     ; SEND ACKNOWLEDGE
                        JSR     @ZOP,2
                        JSR     @ZEP,2  ; PASS ONE CHARACTER

                ; ACCEPT MESSAGE TEXT

LOOP:           JSR     @ZEP,2  ; ACCEPT A CHARACTER
                        LDA 0   .177    ; DATA BYTE MASK
                        AND     1,0     ; EXTRACT DATA
                        STA 0   SAVE    ; AND SAVE IT

                        SUBZL   0,0     ; DISK = 1
                        END             ; RESERVE IT
                        LDA 3   DDOS    ; SYSTEM TABLE
                        LDA 2   BUFF    ; BUFFER ADDRESS
                        LDA 0   SAVE    ; RECOVER DATA
                        JSR     @D.OUT,3; WRITE IT OUT
                        LDA 2   2       ; RECOVER FIT
                        SUBZL   0,0     ; DISK = 1
                        DEO             ; RELEASE IT

                        LDA 0   SAVE    ; RECOVER DATA
                        LDA 1   EOT     ; EOT CHARACTER
                        SEQ     0,1     ; IF NOT EQUAL,
                        JMP     LOOP    ; REPEAT

                ; WRITE END-FILE AND CLOSE

                        SUBZL   0,0     ; DISK = 1
                        END             ; RESERVE IT
                        LDA 3   DDOS    ; SYSTEM TABLE
                        LDA 2   BUFF    ; BUFFER ADDRESS
                        JSR     @D.CYS,3; CALL OVERLAY
                        2.WEF           ; TO WRITE END-FILE
                        JSR     @D.CYS.3; CALL IT AGAIN
                        0.CLO           ; TO CLOSE FILE
                        SUBZL   0,0     ; DISK = 1
                        DEO             ; RELEASE IT

                ; DEALLOCATE BUFFER SPACE

                        LDA 2   .IOC2   ; IOCB ADDRESS
                        .D              ; DEALLOCATE
                        JMP             ; ALWAYS SKIPS
```



```
                A I D S    L E A D S    L I N E    R E C E I V E
02/20/74
                        SUBZR   2,2     ; SETS HIGH BIT
                        STA 0   LRF     ; FILE NOT EMPTY
                        LDA 2   2       ; RECOVER TIT
                        JMP     @.LDS   ; RETURN TO CONTROL

BUFF:           0
SAVE:           0
ALL:            "D
EOT:            "D377
ACK:            -4

.40:            40
.177:           177
.FWB:           T.FW
.LDS:           LEADS

.IOC1:          .+1
                0
                0
BLK1:           0
                0
                T.FW+B.LEN+1

.IOC2:          .+1
                0
                1
BLK2:           0

FILE:           .+1
.TXTM           1
.TXT            *AIDS.LEADS.RX*

                .END
```

A I D C L E A D S L I N E P E C E I V E

[illegible]	[illegible]	[illegible]				
[illegible]	3:1[illegible]	2:15				
[illegible]	3:[illegible]	1:32				
[illegible]	3:2[illegible]	1:45				
[illegible]	3:27	1:46				
[illegible]	3:[illegible]	1:49	2:29	2:46		
[illegible]	1:17	1:21				
[illegible]	1:[illegible]	1:23				
[illegible]	1:[illegible]	1:32				
[illegible]	3:[illegible]	2:37				
[illegible]	3:[illegible]	1:[illegible]				
[illegible]	1:12	2:15				
[illegible]	1:02	1:[illegible]				
[illegible]	2:21	2:32				
[illegible]	1:[illegible]	1:[illegible]	3:02			
[illegible]	1:[illegible]	1:14				
[illegible]	1:3[illegible]	1:2[illegible]				
[illegible]	3:[illegible]	1:[illegible]	2:17	2:24	2:30	2:36
[illegible]	1:1[illegible]	1:17	2:17	2:21		
[illegible]	1:[illegible]	2:15				
[illegible]123[illegible]	2:1[illegible]	1:13	2:22			
[illegible]	3:12	1:2[illegible]				
[illegible]	2:[illegible]	1:[illegible]				
[illegible]	2:17	1:41				
[illegible]	2:[illegible]	2:[illegible]				
[illegible]	2:1[illegible]	1:[illegible]	3:04			

[illegible]

LEADS LINE TRANSMIT (LTX.)

The LEADS line transmit routine, called by the LEADS line control routine, will acknowledge all invitations to transmit bearing the proper drop-code and if an outgoing message is in the "AIDS.LEADS.TX" file, it will be sent. Control is returned to the LEADS line control routine.


```
                        .HEAD   A I D S    L E A D S    L I N E    T R A N S M I T
                        .TITLE  LTX.
000005          DROP=   5                       ; LEADS LINE DROP-CODE
                        .EXTD   CCTR,CCTE       ; CONTROL TABLE LIMITS
                        .EXTD   LTF             ; LEADS OUTPUT FILE POINTERS
                        .EXTD   ZIP,ZOP         ; INPUT/OUTPUT LINE CONTROL
                        .EXTN   LEADS           ; CONTROL LOOP ENTRY POINT
                        .ENT    LTX.
                        .NREL
00000'007401$   LTX.:   JSR     @ZIP,2          ; WAIT FOR AVAILABLE BUFFER
00001'002557            LDA 0   .177            ; MASK OFF LOW BITS
00002'012340            AND     1,0             ; EXTRACT DROP-CODE
00003'034001$           LDA 3   CCTE            ; CONTROL TABLE END
00004'000405    LOOK:   LDA 1   DROP,3          ; LEADS LINE DROP-CODE
00005'125401            SNE     0,1             ; IF THEY MATCH
00006'000417            JMP     OURS            ; COMMUNICATE WITH THEM
00007'002550            LDA 1   .4              ; TABLE ENTRY SIZE
00010'137777            DSZ     1,3             ; ADJUST TABLE POINTER
00011'024002$           LDA 1   CCTR            ; CONTROL TABLE START
00012'136512            SUB     1,3             ; IF AT START OF TABLE,
00013'000771            JMP     LOOK            ; TRY TO MATCH IT
00014'000417            JMP     .+DNS           ; RETURN TO CONTROL LOOP
00015'002033$   OURS:   LDA 1   LTF             ; LEADS OUTPUT FILE
00016'125113            MOVL#   1,1 SNC         ; IF FILE EMPTY
00017'000527            JMP     SACK+1          ; SEND ACK AND QUIT
00020'040532            STA 1   SAVE            ; SAVE FILE POINTER
00021'030530            LDA 2   BUFF            ; BUFFER ADDRESS
00022'151414            MOV#    2,2 SZR         ; IF 0 BUFFER,
00023'000436            JMP     TRY             ; COMMUNICATE WITH THEM

                ; ALLOCATE BUFFER SPACE

00024'000402            SKIP                    ; SKIP AROUND
00025'000444            IOLK                    ; WAIT AWHILE
00026'030536            LDA 2   .IOCI           ; IOCS ADDRESS
00027'000444            .ID                     ; ALLOCATE
00030'000775            JMP     .-3             ; KEEP TRYING

00031'000436            LDA 1   BLK1            ; LOCK ADDRESS
00032'040543            STA 1   BLK2            ; BUFFER ADDRESS
00033'002527            LDA 0   .FNB            ; FNB = 1
00034'101776            ADD     0,1             ; BUFFER ADDRESS + FNB
00035'044541            STA 1   BUFF            ; SAVE FOR LATER

                ; SETUP FILE FOR INPUT

00036'102521            SUBZL   1,1             ; FNB = 1
```

```
                        ENQ             ; RESERVE IT
                        LDA 3   DDOS    ; SYSTEM TABLE
                        LDA 2   FILE    ; FILE NAME
                        JSR     @D.SRC,3; LOCATE FILE
                        JMP     EOF     ; NOT FOUND

                        LDA 0   F.FST,2 ; HIGH-ORDER ADDRESS
                        LDA 1   F.BFP,2 ; LOW-ORDER ADDRESS
                        LDA 3   .377    ; HALF-WORD MASK
                        MOVZR   0,0     ; SHIFT HIGH-ORDER, THEN
                        ADDL    3,1     ; LOW-ORDER, WITH CARRY

                        LDA 2   BUFF    ; BUFFER ADDRESS
                                3,3
                        LDA 3   B.PPP,2 ; CLEAR PROPERTIES
                        ADC     3,3
                        STA 3   B.SS0,2 ; INITIALIZE SSN
                        STA 3   B.SS1,2
                        LDA 3   TOP     ; INITIAL POSITION
                        JMP     GO      ; READ FIRST SECTOR

                ; COMPARE SELECT TO FILE

TEST:   SUBZL   0,0     ; DISK = 1
                ENQ             ; RESERVE IT
                LDA 0   B.DA0,2 ; DISK ADDRESS
                LDA 1   B.DA1,2 ; LOW AND HIGH
                LDA 3   B.NBD,2 ; CURRENT POSITION

GO:             STA 3   SPOT    ; SAVE FOR LATER
                LDA 3   DDOS    ; SYSTEM TABLE
                JSR     @D.RED,3; READ IN SECTOR
                JMP     EOF     ; TREAT ERROR AS EOF
                LDA 3   DDOS    ; SYSTEM TABLE
                JSR     @D.INX,3; FINISH UP
                NOP             ; ALWAYS SKIPS

                LDA 0   SPOT    ; RECOVER POSITION
                STA 0   B.NBD,2 ; BYTE DISPLACEMENT
                LDA 3   DDOS    ; SYSTEM TABLE
                JSR     @D.IN,3 ; READ A CHARACTER
                JMP     EOF     ; OOPS, END-OF-FILE

                LDA 1   .177    ; DATA BYTE MASK
                AND     0,1     ; SAVE DATA BYTE
                SUBZL   0,0     ; DISK = 1
                DEQ             ; RELEASE IT

                LDA 0   SAVE    ; SELECT CHARACTER
                SNE     0,1     ; IF THEY MATCH
                JMP     LOOP    ; SEND THE MESSAGE
                LDA 2   BUFF    ; BUFFER ADDRESS
                DSZ     B.NBD,2 ; SET FOR RETRY
                                ; IN CASE OF SKIP
                JMP     SACK    ; SEND ACK AND QUIT

                ; TRANSMIT MESSAGE TEXT
```



```
02/23/74
 1  00115'102520  LOOP:  SUBZL   0,0     ; DISK = 1
 2  00116'004050         ENQ             ; RESERVE IT
 3  00117'030432         LDA 2   BUFF    ; BUFFER ADDRESS
 4  00120'034041         LDA 3   DDOS    ; SYSTEM TABLE
 5  00121'047404         JSR     @D.IN,3 ; READ A CHARACTER
 6  00122'[illegible]    JMP     EOF     ; [illegible]
 7
 8  00123'024435         LDA 1   .177    ; DATA BYTE MASK
 9  00124'107400         AND     0,1     ; SAVE DATA BYTE
10  00125'102520         SUBZL   0,0     ; DISK = 1
11  00126'004052         DEQ             ; RELEASE IT
12  00127'030002         LDA 2   2       ; RECOVER TIT
13  00130'007005$        JSR     @ZOP,2  ; TRANSMIT DATA
14
15  00131'020424         LDA 0   EOT     ; EOT CHARACTER
16  00132'136414         SEQ     0,1     ; IF NOT EQUAL,
17  00133'000762         JMP     LOOP    ; REPEAT
18  00134'002427         JMP     @.LDS   ; RETURN TO CONTROL
19
20                  ; DEALLOCATE BUFFER SPACE
21
22  00135'102520  EOF:   SUBZL   0,0     ; DISK = 1
23  00136'004052         DEQ             ; RELEASE IT
24  00137'030433         LDA 2   .IOC2   ; IOCL POINTER
25  00140'006044         .IO             ; DEALLOCATE
26  00141'000001         NOP             ; RETURN ADDR
27  00142'102400         SUB     0,0
28  00143'040003$        STA 0   LTF     ; SET IS EOF
29  00144'[illegible]    STA 0   BUFF    ; [illegible]
30
31                  ; ACKNOWLEDGE AND EXIT
32
33  00145'030002  SACK:  LDA 2   2       ; RECOVER TIT
34  00146'024410         LDA 1   ACK     ; ACKNOWLEDGE
35  00147'007005$        JSR     @ZOP,2
36  00150'002413         JMP     @.LDS   ; RETURN TO CONTROL
37
38  00151'000000  BUFF:  0
39  00152'000000  SAVE:  0
40  00153'000000  SPOT:  0
41  00154'000031  TOP:   19.
42  00155'000004  EOT:   "D&77
43  00156'177774  ACK:   -4
44
45
46  00157'000040  .40:   40
47  00160'000177  .177:  177
48  00161'000377  .377:  377
49  00162'000010  .FWB:  T.FW
50  00163'177777  .LDS:  LEADS
51
52  00164'000165' .IOC1: .+1
53  00165'000000         0
54  00166'000000         0
55  00167'000000  BLK1:  0
56  00170'000000         0
57  00171'000424         T.FW+B.LEN+1
58
```

```
                .IOC2:  .+1
                        .
                        .
                        .IOC:

                FILE:   .+1
                        .TXT"   1
                        .TXT    <LEADS.LEADS.TX*
                        .
                        .
                        .
```

TICKET FORMAT (FORM.)

This is the ticket format. Format blocks are described in general in a separate section.


```
                        .HEAD   A I D S    T I C K E T    F O R M A T
                        .TITLE  FORM.
100000  LAST=    100000  ; LAST FIELD FLAG
010000  ADV=     70000   ; ADVANCE-STOP
003000  TAB=     30000   ; TAB-STOP
001000  REV=     10000   ; REVERSE-STOP
000400  CONT=    4000    ; CONTINUE
000200  HBRT=    2000    ; HEADER BRITE
000100  DDIM=    1000    ; DATA DIM
000040  CDIM=    400     ; CARD TRANS CODE
000300  DS3=     300     ; DATA SPACE 3
000200  DS2=     200     ; DATA SPACE 2
000100  DS1=     100     ; DATA SPACE 1
000060  DT3=     60      ; DATA TAB 3 -
000040  DT2=     40      ; DATA TAB 2 /
000020  DT1=     20      ; DATA TAB 1 :
000014  HS3=     14      ; HEADER SPACE 3
000014  HS2=     10      ; HEADER SPACE 2
000004  HS1=     4       ; HEADER SPACE 1
000003  HT3=     3       ; HEADER TAB 3 -
000002  HT2=     2       ; HEADER TAB 2 /
000001  HT1=     1       ; HEADER TAB 1 :
                        .ENT    FORM.
                        .NREL
                        .EXT    FIN,FIC,FIL,FIP
                        .EXT    FIT,FOB,FDB,FOC
                        .EXT    FBB,FDB,FUN,FON
                        .EXT    FBA,FOC,FPO,FPT
                        .EXT    FFP
                        .EXTN   PAGE.
000004  FFP=     4
000000'000001' FORM.:   .LOC    FORM.+FFP
000004'177777           PAGE.   ; PAGE ADVANCE
000005'000015   .FIN:   FORM-TOP/4 ; -23 TICKET
000006'000034   .FFP:   FROM-TOP/4 ; -22 FROM POST
                        .LOC    FORM.+5.S1
000007'000030'          TOP     ; START ADDRESS
```

```
                ;       FIELD INDICES

[illegible]     .FFN:   FILE-TOP/4      ; -20 FILE NUMBER
[illegible]     .FPD:   DEPT-TOP/4      ; -17 DEPARTMENT
[illegible]     .FDC:   DONE-TOP/4      ; -16 DISPOSITION
[illegible]     .FEX:   ASST-TOP/4      ; -15 ASSIST UNITS

[illegible]     .FOW:   TEAM-TOP/4      ; -14 OFFICER(S)
[illegible]     .FUN:   UNIT-TOP/4      ; -13 UNIT NUMBER
[illegible]     .FDB:   DISP-TOP/4      ; -12 DISPATCHED BY
[illegible]     .FRB:   RECD-TOP/4      ; -11 RECEIVED BY

[illegible]     .F24:   COMP-TOP/4      ; -10 COMPLETED TIME
[illegible]     .F23:   ARRV-TOP/4      ; -7  ARRIVED TIME
[illegible]     .F25:   ASSN-TOP/4      ; -6  ASSIGNED TIME
[illegible]     .FIT:   [illegible]     ; -5  INCIDENT TIME

[illegible]     .FIP:   [illegible]     ; -4  INCIDENT POST
[illegible]     .FIL:   SPOT-TOP/4      ; -3  INCIDENT LOCATION
[illegible]     .FIC:   CODE-TOP/4      ; -2  INCIDENT CODE
[illegible]     .FIN:   SORT-TOP/4      ; -1  INCIDENT NATURE

[illegible]     TOP:

    177777  FIN=    .FIN-TOP        ; INCIDENT NATURE
    [illegible]  FIC=    .FIC-TOP        ; INCIDENT CODE
    177775  FIL=    .FIL-TOP        ; INCIDENT LOCATION
    [illegible]  FIP=    .FIP-TOP        ; INCIDENT POST

    177773  FIT=    .FIT-TOP        ; INCIDENT TIME
    177772  F25=    .F25-TOP        ; ASSIGNED TIME
    177771  F23=    .F23-TOP        ; ARRIVED TIME
    177770  F24=    .F24-TOP        ; COMPLETED TIME

    [illegible]  FRB=    .FRB-TOP        ; RECEIVED BY
    [illegible]  FDB=    .FDB-TOP        ; DISPATCHED BY
    [illegible]  FUN=    .FUN-TOP        ; UNIT NUMBER
    [illegible]  FOW=    .FOW-TOP        ; OFFICER(S)

    [illegible]  FEX=    .FEX-TOP        ; ASSIST UNITS
    [illegible]  FDC=    .FDC-TOP        ; DISPOSITION
    [illegible]  FPD=    .FPD-TOP        ; DEPARTMENT
    [illegible]  FFN=    .FFN-TOP        ; FILE NUMBER

    177756  FFP=    .FFP-TOP        ; FROM POST
    177755  FTN=    .FTN-TOP        ; TICKET
```



```
 1      000006  S00=    6.      ; EVENT FIELD SIZE
 2      000010  S01=    8.      ; FILE NUMBER
 3      000004  S02=    4.      ; TIME
 4      000003  S03=    3.      ; ZONE
 5      000002  S04=    2.      ; DAY
 6      000003  S05=    3.      ; MONTH
 7      000002  S06=    2.      ; YEAR
 8      000014  S07=    4.      ; CODE
 9
10      000030  S10=    24.     ; INCIDENT
11      000030  S11=    24.     ; LOCATION
12      000020  S12=    16.     ; CITY
13      000002  S13=    2.      ; STATE
14      000002  S14=    2.      ; DEPT
15      000004  S15=    4.      ; POST
16
17      000030  S20=    24.     ; [illegible]
18      000004  S21=    4.      ; [illegible]
19      000004  S22=    4.      ; [illegible]
20      000004  S23=    4.      ; [illegible]
21      000004  S24=    4.      ; [illegible]
22      000004  S25=    4.      ; [illegible]
23      000004  S26=    4.      ; [illegible]
24      000030  S27=    24.     ; [illegible]
25
26      000030  S30=    24.     ; [illegible]
27      000030  S31=    24.     ; ADDRESS
28      000020  S32=    16.     ; CITY
29      000002  S33=    2.      ; STATE
30      000007  S34=    7.      ; TELEPHONE
31      000003  S35=    3.      ; DISPOSITION
32      000004  S36=    4.      ; FROM POST
33      000030  S37=    24.     ; VICTIM
34
35      000110  S40=    72.     ; REMARKS
```

```
                .1.=    T.[illegible]

[illegible]     [illegible]=    .1.     ; EVENT FIELD
                [illegible]=    [illegible]     ; FILE NUMBER
                [illegible]=    D01+S01 ; TIME
                [illegible]=    D02+S02 ; ZONE
                [illegible]=    D03+S03 ; DAY
                [illegible]=    [illegible]     ; [illegible]
                [illegible]=    [illegible]     ; [illegible]
                [illegible]=    [illegible]     ; [illegible]
                .1.=    [illegible]

                [illegible]=    .1.     ; LAST UNIT
                [illegible]=    D11+S11 ; LOCATION
                [illegible]=    D12+S12 ; CITY
                [illegible]=    D13+S13 ; STATE
                [illegible]=    D14+S14 ; [illegible]
                [illegible]=    D15+S15 ; POST
                .1.=    [illegible]

                [illegible]=    .2.     ; [illegible]
                [illegible]=    D20+S20 ; UNIT
                [illegible]=    [illegible]     ; ASSIGNED
                [illegible]=    [illegible]     ; [illegible]
                [illegible]=    D23+S23 ; [illegible]
                [illegible]=    [illegible]     ; RECEIVED BY
                [illegible]=    [illegible]     ; DISPATCHED BY
                [illegible]=    [illegible]     ; ASSIST
                .1.=    [illegible]

                [illegible]=    .3.     ; [illegible]
                [illegible]=    [illegible]     ; [illegible]
                [illegible]=    [illegible]     ; CITY
                [illegible]=    [illegible]     ; STATE
                [illegible]=    [illegible]     ; TELEPHONE
                D35=    [illegible]     ; DISPOSITION
                D36=    [illegible]     ; FROM POST
                D37=    [illegible]     ; VICTIM
                .4.=    D37+D37

                [illegible]=    .4.     ; REMARKS
                D41=    [illegible]
                [illegible]=    [illegible]
                .5.=    [illegible]
```



```
1                       ; INCIDENT FIELD
2
3   [illegible]  SORT:   ADV+UT1+US2+DS2
4   [illegible]          0*400+11.          ; LINE 1, COL. 12
5   00032/110010         U01*200+L01        ; HEADER OFFSET, SIZE
6   00033/022430         D10*200+S10        ; DATA OFFSET, SIZE
7
8                       ; TIME FIELD
9
10  [illegible]  TIME:   TAB+DS1            ; 1 SPACE AFTER
11  00035/000045         [illegible]*37.    ; LINE 1, COL. 32
12  [illegible]          0                  ; NO HEADER
13  [illegible]          [illegible]        ; DATA OFFSET, SIZE
14
15                      ; TIME ZONE
16
17  [illegible]  ZONE:   COUNT+POINT+DS2    ; 2 SPACES AFTER
18  [illegible]          0*400+42.          ; LINE 1, COL. 43
19  00042/000000         0                  ; NO HEADER
20  [illegible]          [illegible]        ; DATA OFFSET, SIZE
21
22                      ; DAY FIELD
23
24  [illegible]  DATE:   TAB+COUNT+DS1      ; 1 SPACE AFTER
25  [illegible]          0*400+47.          ; LINE 1, COL. 48
26  [illegible]          0                  ; NO HEADER
27  [illegible]          [illegible]        ; DATA OFFSET, SIZE
28
29                      ; MONTH FIELD
30
31  00050/004100         COUNT+DS1          ; 1 SPACE AFTER
32  00051/000062         0*400+50.          ; LINE 1, COL. 51
33  [illegible]          0                  ; NO HEADER
34  [illegible]          D05*200+S05        ; DATA OFFSET, SIZE
35
36                      ; YEAR FIELD
37
38  00054/006240         COUNT+NUMBR+DS2    ; 2 SPACES AFTER
39  [illegible]          [illegible]*55.    ; LINE 1, COL. 57
40  [illegible]          H19*200+2          ; HEADER "19"
41  00057/021002         D06*200+S06        ; DATA OFFSET, SIZE
42
43                      ; INCIDENT CODE
44
45  [illegible]  CODE:   TAB+DS1            ; 1 SPACE BEFORE
46  00061/000101         [illegible]*65.    ; LINE 1, COL. 66
47  00062/112004         [illegible]        ; HEADER OFFSET, SIZE
48  [illegible]          [illegible]        ; DATA OFFSET, SIZE
```

```
                ; FILE NUMBER #1

                FILE:   ...                     ; 2 SPACES BEFORE
                        ...*400+71.             ; LINE 1, COL. 72
                        0                       ; NO HEADER
                        D41*200+3               ; DATA OFFSET, SIZE

                ; FILE NUMBER #2

                        ...                     ; - BEFORE
                        ...+75.                 ; LINE 1, COL. 76
                        0                       ; NO HEADER
                        D...*200+5              ; DATA OFFSET, SIZE
```



```
                ; LOCATION FIELD

0007400702 11   SPOT:   ADV+HT1+HS2+DS2
00075000413             1*400+11.       ; LINE 2, COL. 12
00076113010             H13*2...+L13    ; HEADER OFFSET, SIZE
00077030430             D11*20...+S11   ; DATA OFFSET, SIZE

                ; CITY FIELD

00100030244     CITY:   TA...+DT2+DS2   ; 2 SPACES AFTER
00101000445             1*400+37.       ; LINE 2, COL. 38
00102000000             0               ; NO HEADER
00103030424             D12*2...+S12    ; DATA OFFSET, SIZE

                ; STATE FIELD

00104034200             TA...+COUNT+DS2 ; 2 SPACES AFTER
00105...3...            1*400+56.       ; LINE 2, COL. 57
00106000000             0               ; NO HEADER
00107...2...            D13*2...+S13    ; DATA OFFSET, SIZE

                ; POST FIELD

00110030004     POST:   TAB+HS1         ; 1 SPACE BEFORE
00111000501             1*400+65.       ; LINE 2, COL. 66
00112...1...            H04*200+L04     ; HEADER OFFSET, SIZE
00113043404             D15*200+S15     ; DATA OFFSET, SIZE

                ; DEPT FIELD

00114001010     DEPT:   ...             ; 2 SPACES BEFORE
00115000507             1*400+71.       ; LINE 2, COL. 72
00116000000             0               ; NO HEADER
00117043002             D14*200+S14     ; DATA OFFSET, SIZE

                ; EVENT FIELD

00120...        FORM:   ...             ; 2 SPACES BEFORE
00121...513             1*400+75.       ; LINE 2, COL. 76
00122...                0               ; NO HEADER
00123...5               D...*200+5      ; DATA OFFSET, SIZE
```

```
                A I D S    T I C K E T    F O R M A T

                ; CALLER FIELD

                        ADV+HT1+HS2+DS2
                        2*400+11.       ; LINE 3, COL. 12
                        H05*200+L05     ; HEADER OFFSET, SIZE
                        D30*200+S30     ; DATA OFFSET, SIZE

                ; VICTIM FIELD

                        TAB+HT1+HS2     ; 2 SPACES BEFORE
                        2*400+46.       ; LINE 3, COL. 47
                        H08*200+L08     ; HEADER OFFSET, SIZE
                        D37*200+S37     ; DATA OFFSET, SIZE

                ; DISPOSITION

                DONE:   TAB+HT1         ; 1 SPACE BEFORE
                        2*400+77.       ; LINE 3, COL. 78
                        H09*200+L09     ; HEADER OFFSET, SIZE
                        D35*200+S35     ; DATA OFFSET, SIZE
```



```
12/20/74                A I D S    T I C K E T    F O R M A T

                ; ADDRESS FIELD

                        ADV+HT1+HS2+DS2 ; 2 SPACES BEFORE
                        3*400+11.       ; LINE 4, COL. 12
                        H06*200+L06     ; HEADER OFFSET, SIZE
                        D31*200+S31     ; DATA OFFSET, SIZE

                ; CITY FIELD

                        TAB+DT2+DS2     ; , 2 SPACES AFTER
                        3*400+37.       ; LINE 4, COL. 38
                        0               ; NO HEADER
                        D32*200+S32     ; DATA OFFSET, SIZE

                ; STATE FIELD

                        TAB+DT2+DS2     ; 2 SPACES BEFORE
                        3*400+56.       ; LINE 4, COL. 57
                        0               ; NO HEADER
                        D33*200+S33     ; DATA OFFSET, SIZE

                ; TELEPHONE #1

                        TAB+HT1+HS2     ; 2 SPACES BEFORE
                        3*400+72.       ; LINE 4, COL. 73
                        H07*200+L07     ; HEADER OFFSET, SIZE
                        D34*200+3       ; DATA OFFSET, SIZE

                ; TELEPHONE #2

                        COUT+HBRT+HT3   ; - SPACE 4, COL. 77
                        3*400+76.       ; LINE 4, COL. 77
                        0               ; NO HEADER
                        D34+3*200+4     ; DATA OFFSET, SIZE
```

```
                        ; ASSISTING

00214/070204    ASST:   ADV+HS1+DS2     ; 1 SPACE BEFORE, 2 AFTER
00215/002422            5*400+18.       ; LINE 6, COL. 19
00216/140406            H15*200+L15     ; HEADER OFFSET, SIZE
00217/060430            D27*200+S27     ; DATA OFFSET, SIZE

                        ; RECEIVED BY

00220/030204    RECD:   TAB+HS1+DS2     ; 1 SPACE BEFORE, 2 AFTER
00221/002473            5*400+60.       ; LINE 6, COL. 57
00222/142413            H16*200+L16     ; HEADER OFFSET, SIZE
00223/060414            D28*200+S28     ; DATA OFFSET, SIZE

                        ; DISPATCHED BY

00224/030204    DISP:   TAB+HS1         ; 1 SPACE BEFORE
00225/002514            5*400+76.       ; LINE 6, COL. 77
00226/144015            H17*200+L17     ; HEADER OFFSET, SIZE
00227/057404            D29*200+S29     ; DATA OFFSET, SIZE

                        ; REMARKS

00230/070011    NOTE:   ADV+HT1+HS2
00231/003010            1*400+8.        ; LINE 7, COL. 9
00232/126015            H18*200+L18     ; HEADER OFFSET, SIZE
00233/120511            D30*200+S30     ; DATA OFFSET, SIZE

00234/034011            TAB+CONT+HS2
00235/003411            1*400+8.        ; LINE 7, COL. 9
00236/000000            0               ; NO HEADER
00237/142510            D31*200+S31     ; DATA OFFSET, SIZE

00240/034011            TAB+CONT+HS2
00241/004215            1*400+8.        ; LINE 7, COL. 9
00242/000000            0               ; NO HEADER
00243/104511            D32*200+S32     ; DATA OFFSET, SIZE

00244/130000            LAST+745        ; LAST FIELD
00245/005000            12*400          ; LINE 11, COL. 1
00246/000000            0               ; NO HEADER
00247/000000            0               ; NO DATA
```

```
                A I D S    T I C K E T    F O R M A T

                ; HEADERS

                H0=     8.          ; INCIDENT HEADER LENGTH
                L0=     4.          ; CODE
                L1=     8.          ; LOCATION
                L2=     4.          ; POST
                L3=     8.          ; CALLER
                L4=     8.          ; ADDRESS
                L5=     2.          ; TELEPHONE
                L6=     6.          ; VICTIM
                L7=     4.          ; DISPOSITION

                L8=     4.          ; UNIT
                L9=     8.          ; ASSIGNED
                L10=    7.          ; ARRIVED
                L11=    2.          ; COMPLETED
                L12=    4.          ; FROM
                L13=    6.          ; ASSIST
                L14=    11.         ; RECEIVED BY
                L15=    13.         ; DISPATCHED BY
                L16=    5.          ; NOTES
                L17=    2.          ; "19"

                H1=     .-TOP*2     ; INCIDENT HEADER
                H2=     H1+L1       ; CODE
                H3=     H2*L2       ; LOCATION
                H4=     H3+L3       ; POST
                H5=     H4+L4       ; CALLER
                H6=     H5+L5       ; ADDRESS
                H7=     H6+L6       ; TELEPHONE
                H8=     H7+L7       ; VICTIM
                H9=     H8+L8       ; DISPOSITION

                H16=    H9+L9       ; NOTES
                H17=    H16+L16     ; "19"

                H10=    H17+L17     ; UNIT
                H11=    H10+L10     ; ASSIGNED
                H12=    H11+L11     ; ARRIVED
                H13=    H12+L12     ; COMPLETED
                H14=    H13+L13     ; FROM
                H15=    H14+L14     ; ASSIST
                H16=    H15+L15     ; RECEIVED BY
                H17=    H16+L16     ; DISPATCHED BY
                END=    H17+L17

                .TXTM   1
                .TXTN   1
                .TXTC   1

                .TXT    *incidentcodelocationpostcaller  address telephonevictimd
                .TXT    *12unitassignedarrivecomplete fromassistreceived by dispat

                .END
```



```
                A I D S    T I C K E T    F O R M A T
02/20/74

SYMBOL  VALUE   DEF'N  REFERENCES

ADV     070000  1:06   5:03   7:03   8:03   9:03   10:14  11:13  11:14
ARRV    000200  10:24  2:14
ASSN    000174  10:17  2:15
ASST    000214  11:03  2:16
CITY    000100  7:10
COMP    000204  10:31  2:13
CONT    004000  1:10   5:17   5:24   5:31   5:38   6:14   7:17   8:17
                       9:31   11:29  11:34
CRLF    000400  1:13
D00     000052  4:03   4:04   7:41
D01     000060  4:04   4:05   6:06   6:13
D02     000071  4:05   4:06   5:13
D03     000074  4:06   4:07   5:20
D04     000077  4:07   4:08   5:27
D05     000101  4:08   4:09   5:34
D06     000104  4:09   4:10   5:41
D07     000106  4:10   4:11   5:48
D10     000112  4:13   4:14   5:55
D11     000142  4:14   4:15   7:06
D12     000172  4:15   4:16   7:13
D13     000212  4:16   4:17   7:20
D14     000214  4:17   4:18   7:24
D15     000216  4:18   4:19   7:27
        000222  4:21   4:22   10:13
D2      000252  4:22   4:23   10:16
D22     000256  4:23   4:24   10:20
D23     000262  4:24   4:25   10:27
D24     000206  4:25   4:26   10:34
D25     000272  4:26   4:27   11:13
D26     000276  4:27   4:28   11:20
D27     000302  4:28   4:29   11:26
        000332  4:31   4:32   8:06
D31     000362  4:32   4:33   9:06
D32     000412  4:33   4:34   9:13
D33     000432  4:34   4:35   9:20
D34     000434  4:35   4:36   9:27   9:34
D35     000443  4:36   4:37   8:20
D36     000446  4:37   4:38   10:41
D37     000452  4:38   4:39   8:13
        000542  4:41   4:42   11:27
D41     000612  4:42   4:43   11:32
D42     000722  4:43   4:44   11:37
DATE    000044  5:24
DDIM    001000  1:12   5:17   7:31
        000114  7:31   2:14
        000224  11:17  2:17
DONE    000134  8:17   2:16
        000100  1:17   5:14   5:24   5:31
        000200  1:16   5:03   5:17   5:38   7:03   7:14   7:17   8:03
                       9:03   9:10   9:17   10:03  10:14  10:17  11:03
                       10:38  11:03  11:10
        001300  1:15
        000020  1:21
        000040  1:20   7:13   9:13
```

SYMBOL	VALUE	DEF'N	REFERENCES						
L04	000004	12:06	7:26	12:28					
L05	000010	12:07	8:05	12:29					
L06	000010	12:08	9:05	12:30					
L07	000011	12:09	9:26	12:31					
L08	000006	12:10	8:12	12:32					
L09	000004	12:11	9:19	12:34					
L10	000004	12:13	10:05	12:38					
L11	000010	12:14	10:19	12:39					
L12	000007	12:15	10:26	12:40					
L13	000011	12:16	10:33	12:41					
L14	000004	12:17	10:39	12:42					
L15	000006	12:18	11:05	12:43					
L16	000013	12:19	11:12	12:44					
L17	000015	12:20	11:19	12:45					
L18	000005	12:21	11:26	12:35					
L19	000002	12:22	12:37						
L20	100000	1:05	11:29						
L21	000234'	11:24							
L22	'0	1:41	1:45						
L23	000110'	7:24	2:13						
L24	000220'	11:19	2:11						
L25	000004	1:42	1:44						
L26	010000	1:03							
L27	000006	3:01	4:14						
L28	000013	3:02	4:15						
L29	000004	3:03	4:16	9:13					
L30	000003	3:04	4:17	9:28					
L31	000002	3:05	4:18	9:27					
L32	000003	3:06	4:19	9:34					
L33	000002	3:07	4:10	9:31					
L34	000004	3:08	4:11	9:43					
L35	000030	3:10	4:14	9:36					
L36	000034	3:11	4:15	7:36					
L37	000020	3:12	4:16	7:13					
L38	000002	3:13	4:17	7:24					
L39	000002	3:14	4:18	7:34					
L40	000004	3:15	4:19	7:27					
L41	000030	3:17	4:22	10:13					
L42	000004	3:18	4:23	10:16					
L43	000004	3:19	4:24	10:20					
L44	000004	3:20	4:25	10:27					
L45	000004	3:21	4:26	10:24					
L46	000004	3:22	4:27	11:13					
L47	000004	3:23	4:28	11:20					
L48	000031	3:24	4:29	11:16					
L49	000030	3:26	4:32	9:16					
L50	000030	3:27	4:33	9:16					
L51	000020	3:28	4:34	9:13					
L52	000002	3:29	4:35	9:20					
L53	000027	3:30	4:36	9:17					
L54	000003	3:31	4:37	9:17					
L55	000004	3:32	4:38	10:31					
L56	000030	3:33	4:39	9:15					
L57	000110	3:37	4:42	9:13	9:44	11:27	11:32	11:37	
L58	000030'	5:03	2:21						

IS A I R S TICKET FORMAT

[illegible]	[illegible]	REMARKS						
[illegible]	7:03	2:19						
[illegible]	1:37	5:17	5:24	5:45	7:10	7:17	7:24	9:11
		9:17	9:17	9:17	9:24	10:03	10:17	10:24
		10:31	10:33	11:10	11:17	11:29	11:34	11:39
[illegible]	10:17	2:03						
[illegible]	9:17	2:16						
[illegible]	2:00	1:46	1:47	1:49	2:03	2:04	2:05	2:06
		2:08	2:09	2:10	2:11	2:13	2:14	2:15
		2:16	2:18	2:19	2:20	2:21	2:25	2:26
		2:27	2:28	2:30	2:31	2:32	2:33	2:35
		2:36	2:37	2:38	2:40	2:41	2:42	2:43
		2:45	2:46	12:24				
[illegible]	1:17	2:09						
[illegible]	9:1							
[illegible]	2:17							
	4:0	4:03						
	4:11	4:13						
	4:19	4:21						
	4:	4:31						
	4:2	4:41						
	4:3							
[illegible]	2:3	2:32						
[illegible]	2:	2:33						
[illegible]	2:3	2:31						
[illegible]	2:3	2:36						
[illegible]	2:	2:41						
[illegible]	2:	2:42						
[illegible]	2:	2:43						
[illegible]	2:4	2:45						
[illegible]	2:	2:26						
[illegible]	2:2	2:27						
[illegible]	2:2	2:25						
[illegible]	2:	2:28						
[illegible]	2:3	2:30						
[illegible]	2:	2:33						
[illegible]	2:	2:42						
[illegible]	2:3	2:35						
[illegible]	2:4	2:46						
[illegible]	2:3	2:37						

[illegible]: [illegible]

TICKET FORMAT (BACK.)

This the the format block for the back side of a ticket.


```
                        .HEAD   A I D S     A D D - O N     F O R M A T

                        .TITLE  BACK.

                100000  LAST=   100000  ; LAST FIELD FLAG
                070000  ADV=    7[illegible]  ; ADVANCE-STOP
                030000  TAB=    3[illegible]  ; TAB-STOP
                010000  REV=    10[illegible]  ; [illegible]-STOP
                004000  CO[illegible]=  4000  ; [illegible]
                [illegible]  [illegible]=  2[illegible]  ; [illegible]
                [illegible]  [illegible]=  1[illegible]  ; [illegible]
                [illegible]  [illegible]=  4[illegible]  ; [illegible]

                [illegible]  [illegible]=  3[illegible]  ; DATA SPACE 3
                [illegible]  [illegible]=  2[illegible]  ; DATA SPACE 2
                [illegible]  [illegible]=  1[illegible]  ; DATA SPACE 1

                [illegible]  [illegible]=  [illegible]  ; DATA TERM -
                [illegible]  [illegible]=  [illegible]  ; DATA TERM ,
                [illegible]  [illegible]=  2[illegible]  ; [illegible] .

                [illegible]  [illegible]=  14  ; HEADER SPACE 3
                [illegible]  [illegible]=  1[illegible]  ; HEADER SPACE 2
                [illegible]  [illegible]=  1  ; HEADER SPACE 1

                [illegible]  [illegible]=  3  ; HEADER TERM -
                [illegible]  [illegible]=  2  ; HEADER TERM /
                [illegible]  [illegible]=  1  ; HEADER TERM :

                        .[illegible]  [illegible].
                        .[illegible]

                        .[illegible]  [illegible].

                [illegible]  [illegible]=  4

[illegible]     BACK.:  .LOC    BACK.+[illegible]
[illegible]             PA[illegible].  ; PAGE ADVANCE
[illegible]             .LOC    BACK.+[illegible]
[illegible]             TOP     ; START ADDRESS

[illegible]     TOP:
```

```
                ;       WANTED/MISSING PERSON

            S00=    8.      ; WANTED/MISSING
            S01=    24.     ; NAME
            S02=    1.      ; SEX
            S03=    1.      ; RACE
            S04=    2.      ; ORIGIN
            S05=    6.      ; BIRTHDATE
            S06=    2.      ; AGE

            S10=    3.      ; HEIGHT
            S11=    3.      ; WEIGHT
            S12=    3.      ; HAIR COLOR
            S13=    3.      ; EYE COLOR
            S14=    3.      ; SKIN COLOR
            S15=    10.     ; MARKS

            S20=    9.      ; SOCIAL SECURITY
            S21=    2.      ; DLS
            S22=    20.     ; DLN
            S23=    2.      ; DLY
            S24=    9.      ; FBI
            S25=    7.      ; BCI

            S30=    64.     ; MISCELLANEOUS
            S40=    3.      ; DISPOSITION

                ;       WANTED/STOLEN VEHICLE

            S50=    7.      ; COLOR
            S51=    2.      ; YEAR
            S52=    4.      ; MAKE
            S53=    4.      ; MODEL
            S54=    2.      ; STYLE

            S60=    18.     ; VIN
            S61=    2.      ; LIY
            S62=    2.      ; LIS
            S63=    2.      ; LIT
            S64=    8.      ; LIC

            S70=    8.      ; WANTED/STOLEN
            S80=    40.     ; MISCELLANEOUS
```



```
1       000056  .0.=   T.FW+B.FWD+2*2
2
3               ;       WANTED/MISSING PERSON
4
5       000056  D00=    .0.     ; WANTED/MISSING
6       000066  D01=    D00+S00 ; NAME
7       000116  D02=    D01+S01 ; SEX
8       000117  D03=    D02+S02 ; RACE
9       000120  D04=    D03+S03 ; ORIGIN
10      000122  D05=    D04+S04 ; BIRTHDATE
11      000130  D06=    D05+S05 ; AGE
12      000132  .1.=    D06+S06
13
14      000132  D10=    .1.     ; HEIGHT
15      000135  D11=    D10+S10 ; WEIGHT
16      000140  D12=    D11+S11 ; HAIR COLOR
17      000143  D13=    D12+S12 ; EYE COLOR
18      000146  D14=    D13+S13 ; SKIN COLOR
19      000151  .2.=    D14+S14
20
21      000151  D20=    .2.     ; SOCIAL SECURITY
22      000162  D21=    D20+S20 ; DLS
23      000164  D22=    D21+S21 ; DLN
24      000214  D23=    D22+S22 ; DLY
25      000212  D24=    D23+S23 ; FBI
26      000223  D25=    D24+S24 ; BCI
27      000232  .3.=    D25+S25
28
29      000232  D15=    .3.     ; MARKS
30      000244  D30=    D15+S15
31      000344  D31=    D30+S30 ; MISCELLANEOUS
32      000444  D32=    D31+S30
33      000544  .4.=    D32+S30
34      000544  D40=    .4.     ; DISPOSITION
35      000547  .5.=    D40+S40
36
37              ;       WANTED/STOLEN VEHICLE
38
39      000547  D50=    .5.     ; COLOR
40      000556  D51=    D50+S50 ; YEAR
41      000562  D52=    D51+S51 ; MAKE
42      000564  D53=    D52+S52 ; MODEL
43      000570  D54=    D53+S53 ; STYLE
44      000572  .6.=    D54+S54
45
46      000572  D60=    .6.     ; VIN
47      000614  D61=    D60+S60 ; LIY
48      000616  D62=    D61+S61 ; LIS
49      000620  D63=    D62+S62 ; LIT
50      000622  D64=    D63+S63 ; LIC
51      000630  .7.=    D64+S64
52
53      000630  D70=    .7.     ; WANTED/STOLEN
54      000710  .8.=    D70+S70
55              D80=    .8.
56      000712  D81=    D80+S80 ; MISCELLANEOUS
57      001062  .9.=    D81+S81
58      001132  .A.=    D9.+S9.
```

```
                ; WANTED/MISSING

                TAB+HT1+HS1
                ADV+LINE+COL+1.         ; LINE 1, COL. 1
                0                       ; NO HEADER
                D00*200+S00             ; DATA OFFSET, SIZE

                ; DATE FIELD

                ADV+HT1+HS1+DS2
                ADV+LINE+COL+11.        ; LINE 1, COL. 11
                0                       ; NO HEADER
                D01*200+S01             ; DATA OFFSET, SIZE

                ; SEX FIELD

                ADV+HT1+HS1+DS1
                ADV+LINE+COL+42.        ; LINE 1, COL. 42
                HEADER OFFSET*L02       ; HEADER OFFSET, SIZE
                D02*200+S02             ; DATA OFFSET, SIZE

                ; RACE FIELD

                ADV+HT1+HS1+DS1
                ADV+LINE+COL+50.        ; LINE 1, COL. 50
                HEADER OFFSET*L03       ; HEADER OFFSET, SIZE
                D03*200+S03             ; DATA OFFSET, SIZE

                ; HEIGHT FIELD

                ADV+HT1+HS1+DS1
                ADV+LINE+COL+57.        ; LINE 1, COL. 57
                HEADER OFFSET*L04       ; HEADER OFFSET, SIZE
                D04*200+S04             ; DATA OFFSET, SIZE
```



```
1                       ; BIRTH DATE #1
2
3       000034'030005           TAB+HT1+HS1
4       000035'000100           ADV+LINE+COL+1.         ; LINE 1, COL. 1
5       000036'130403           HEADER OFFSET*L05       ; HEADER OFFSET, SIZE
6       000037'024402           D05*200+2               ; DATA OFFSET, SIZE
7
8                       ; BIRTH DATE #2
9
10      000040'006002           ADV+HT2
11      000041'000103           ADV+LINE+COL+67.        ; LINE 1, COL. 67
12      000042'000004           0                       ; NO HEADER
13      000043'026402           D06*200+2               ; DATA OFFSET, SIZE
14
15                      ; BIRTH DATE #3
16
17      000044'006102           ADV+HT2+HS1+DS1
18      000045'000106           ADV+LINE+COL+71.        ; LINE 1, COL. 71
19      000046'000000           0                       ; NO HEADER
20      000047'026402           D07*200+2               ; DATA OFFSET, SIZE
21
22                      ; AGE FIELD
23
24      000050'030005           TAB+HT1+HS1
25      000051'000116           ADV+LINE+COL+78.        ; LINE 1, COL. 78
26      000052'131203           HEADER OFFSET*L08       ; HEADER OFFSET, SIZE
27      000053'026002           D08*200+S08             ; DATA OFFSET, SIZE
```

```
                ; HEIGHT FIELD

                        ADV+HTI+HSI+DS2
                        1*400+05.       ; LINE 2, COL. 6
                        H10*200+L10     ; HEADER OFFSET, SIZE
                        D10*200+S10     ; DATA OFFSET, SIZE

                ; WEIGHT FIELD

                        TAB+HTI+HSI+DS2
                        1*400+15.       ; LINE 2, COL. 16
                        H11*200+L11     ; HEADER OFFSET, SIZE
                        D11*200+S11     ; DATA OFFSET, SIZE

                ; HAIR COLOR

                        TAB+HTI+HSI+DS2
                        1*400+26.       ; LINE 2, COL. 27
                        H12*200+L12     ; HEADER OFFSET, SIZE
                        D12*200+S12     ; DATA OFFSET, SIZE

                ; EYE COLOR

                        TAB+HTI+HSI+DS2
                        1*400+36.       ; LINE 2, COL. 37
                        H13*200+L13     ; HEADER OFFSET, SIZE
                        D13*200+S13     ; DATA OFFSET, SIZE

                ; SKIN COLOR

                        TAB+HTI+HSI+DS2
                        1*400+47.       ; LINE 2, COL. 48
                        H14*200+L14     ; HEADER OFFSET, SIZE
                        D14*200+S14     ; DATA OFFSET, SIZE

                ; FBI NUMBER

                        TAB+HTI+HSI+DS2
                        1*400+57.       ; LINE 2, COL. 58
                        H24*200+L24     ; HEADER OFFSET, SIZE
                        D24*200+S24     ; DATA OFFSET, SIZE

                ; DOB FIELD

                        TAB+HTI+HSI
                        1*400+73.       ; LINE 2, COL. 74
                        H25*200+L25     ; HEADER OFFSET, SIZE
                        D25*200+S25     ; DATA OFFSET, SIZE
```



```
                ; SOCIAL SECURITY #1

                        ADV+HTI+HSI
00111 001005            2*400+05.       ; LINE 3, COL. 6
00112 137403            H20*200+L20     ; HEADER OFFSET, SIZE
00113 032203            D20*200+3       ; DATA OFFSET, SIZE

                ; SOCIAL SECURITY #2

                        CONT+HBRT+HT3+DS2
00115 001011            2*400+09.       ; LINE 3, COL. 10
00116 000000            0               ; NO HEADER
00117 033002            D20+3*200+2     ; DATA OFFSET, SIZE

                ; SOCIAL SECURITY #3

                        CONT+HBRT+HT3+DS2
00120 006203            2*400+12.       ; LINE 3, COL. 13
00121 001014            0               ; NO HEADER
00122 000000            D20+5*200+4     ; DATA OFFSET, SIZE
00123 033404

                ; LICENSE STATE

                        TAB+HTI+HSI+DS2
00125 001027            2*400+23.       ; LINE 3, COL. 24
00126 140203            H21*200+L21     ; HEADER OFFSET, SIZE
00127 034402            D21*200+S21     ; DATA OFFSET, SIZE

                ; LICENSE NUMBER

                        TAB+HTI+HSI+DS2
00131 001040            2*400+32.       ; LINE 3, COL. 33
00132 141003            H22*200+L22     ; HEADER OFFSET, SIZE
00133 035024            D22*200+S22     ; DATA OFFSET, SIZE

                ; LICENSE YEAR

                        TAB+HTI+HSI+DS2
00134 030205            2*400+59.       ; LINE 3, COL. 60
00135 001073            H23*200+L23     ; HEADER OFFSET, SIZE
00136 141603            D23*200+S23     ; DATA OFFSET, SIZE
00137 042002
```

```
                ; SCARS, MARKS, TATTOOS

                TAB+HTI+HSI
                2*4+77.         ; LINE 3, COL. 71
                H2*2**+L2       ; HEADER OFFSET, SIZE
                D2*2**+S2       ; DATA OFFSET, SIZE

                ; [illegible]

                TAB+HTI+HSI+DSI
                3*4**+**.       ; LINE 4, COL. 7
                H3*2**+L3       ; HEADER OFFSET, SIZE
                D3*2**+S3       ; DATA OFFSET, SIZE

                TAB+COMT+HSI+DSI
                4*4**+06.       ; LINE 5, COL. 7
                0               ; NO HEADER
                D31*200+S30     ; DATA OFFSET, SIZE

                TAB+COMT+HSI+DSI
                5*4**+06.       ; LINE 6, COL. 7
                0               ; NO HEADER
                D32*200+S30     ; DATA OFFSET, SIZE

                ; DISPOSITION

                TAB+HTI+HSI
                **4*4**+77.     ; LINE 6, COL. 71
                H**2*2**+L**    ; HEADER OFFSET, SIZE
                D**2*2**+S**    ; DATA OFFSET, SIZE
```



```
                ; COLOR FIELD

00164'030205            TAB+HTI+HSI+DS2
00165'003007            6*4**+07.       ; LINE 7, COL. 8
00166'144005            H5**2**+L5      ; HEADER OFFSET, SIZE
00167'131403            D50*200+3       ; DATA OFFSET, SIZE

                ; YEAR FIELD

00170'030205            TAB+HTI+HSI+DS2
00171'003022            6*4**+18.       ; LINE 7, COL. 10
00172'145204            H5**2**+L5      ; HEADER OFFSET, SIZE
00173'133402            D5**2**+S5      ; DATA OFFSET, SIZE

                ; MAKE FIELD

00174'030205            TAB+HTI+HSI+DS2
00175'003031            6*4**+29.       ; LINE 7, COL. 2*
00176'146204            H52*200+L52     ; HEADER OFFSET, SIZE
00177'134004            D52*200+S52     ; DATA OFFSET, SIZE

                ; MODEL FIELD

00200'030205            TAB+HTI+HSI+DS2
00201'003051            6*4**+41.       ; LINE 7, COL. 42
00202'147205            H53*200+L53     ; HEADER OFFSET, SIZE
00203'135003            D53*200+S53     ; DATA OFFSET, SIZE

                ; STYLE FIELD

00204'030205            TAB+HTI+HSI+DS2
00205'003065            6*4**+53.       ; LINE 7, COL. 55
00206'150405            H54*2**+L54     ; HEADER OFFSET, SIZE
00207'136002            D54*2**+S54     ; DATA OFFSET, SIZE
```

```
                ; LICENSE YEAR

                        TAB+ITI+HSI
                        6*400+63.       ; LINE 7, COL. 64
                        H61*200+L61     ; HEADER OFFSET, SIZE
                        D61*200+S61     ; DATA OFFSET, SIZE

                ; LICENSE STATE

                        TAB+HSI
                        6*400+66.       ; LINE 7, COL. 67
                        0               ; NO HEADER
                        D62*200+S62     ; DATA OFFSET, SIZE

                ; LICENSE TYPE

                        TAB+HSI
                        6*400+69.       ; LINE 7, COL. 70
                        0               ; NO HEADER
                        D63*200+S63     ; DATA OFFSET, SIZE

                ; LICENSE NUMBER

                        TAB+HSI
                        6*400+72.       ; LINE 7, COL. 73
                        0               ; NO HEADER
                        D64*200+S64     ; DATA OFFSET, SIZE
```



```
                ; WANTED/STOLEN

                        REV+DDIM+DS2
00230'011200            7*400+00.       ; LINE 8, COL. 1
00231'003400            0               ; NO HEADER
00232'000000            D70*200+S70     ; DATA OFFSET, SIZE
00233'146410

                ; IDENTIFICATION

                        ADV+ITI+HSI+DS2
00234'070205            7*400+14.       ; LINE 8, COL. 15
00235'003416            H71*200+L71     ; HEADER OFFSET, SIZE
00236'151452            D71*200+S71     ; DATA OFFSET, SIZE
00237'136432

                ; MISCELLANEOUS

00240'003405            TAB+ITI+HSI
00241'003450            7*400+40.       ; LINE 8, COL. 41
00242'124404            H80*200+L80     ; HEADER OFFSET, SIZE
00243'150450            D80*200+S80     ; DATA OFFSET, SIZE

00244'034000            TAB+CONT
00245'004050            8.*400+40.      ; LINE 9, COL. 41
00246'000000            0               ; NO HEADER
00247'152451            D81*200+S81     ; DATA OFFSET, SIZE

00250'034000            TAB+CONT
00251'004450            9.*400+40.      ; LINE 10, COL. 41
00252'000000            0               ; NO HEADER
00253'174450            D82*200+S80     ; DATA OFFSET, SIZE

00254'134410            LNS+TYP
00255'005400            10.*400+0       ; LINE 11, COL. 1
00256'000000            0               ; NO HEADER
00257'000000            0               ; NO DATA
```

```
          ; RACE=B3

          L00=    4.      ; MISC.
          L01=    4.      ; DISP.
          L02=    3.      ; SEX
          L03=    4.      ; RACE
          L04=    3.      ; POB
          L05=    3.      ; DOB
          L06=    3.      ; AGE

          L10=    3.      ; WEIGHT
          L11=    3.      ; HEIGHT
          L12=    4.      ; HAIR
          L13=    3.      ; EYES
          L14=    4.      ; SKIN
          L15=    5.      ; MARKS

          L20=    3.      ; SOCIAL SECURITY
          L21=    3.      ; DLS
          L22=    3.      ; DLN
          L23=    3.      ; DLY
          L24=    3.      ; FBI
          L25=    3.      ; BCI

          L50=    5.      ; COLOR
          L51=    4.      ; YEAR
          L52=    4.      ; MAKE
          L53=    5.      ; MODEL
          L54=    5.      ; STYLE
          L60=    2.      ; ID #
          L61=    3.      ; LICENSE
```



```
1         000520  H00=    .-TOP*2 ; MISC.
2         000524  H01=    H00+L00 ; DISP.
3         000530  H02=    H01+L01 ; SEX
4         000533  H03=    H02+L02 ; RACE
5         000537  H04=    H03+L03 ; POB
6         000542  H05=    H04+L04 ; DOB
7         000545  H06=    H05+L05 ; AGE
8
9         000554  H10=    H06+L06 ; WEIGHT
10        000563  H11=    H10+L10 ; HEIGHT
11        000566  H12=    H11+L11 ; HAIR
12        000562  H13=    H12+L12 ; EYES
13        000565  H14=    H13+L13 ; SKIN
14        000571  H15=    H14+L14 ; MARKS
15
16        000576  H20=    H15+L15 ; SOCIAL SECURITY
17        000601  H21=    H20+L20 ; DLS
18        000604  H22=    H21+L21 ; DLN
19        000607  H23=    H22+L22 ; DLY
20        000612  H24=    H23+L23 ; FBI
21        000615  H25=    H24+L24 ; BCI
22
23        000621  H50=    H25+L25 ; COLOR
24        000625  H51=    H50+L50 ; YEAR
25        000631  H52=    H51+L51 ; MAKE
26        000635  H53=    H52+L52 ; MODEL
27        000641  H54=    H53+L53 ; STYLE
28        000647  H60=    H54+L54 ; ID #
29        000651  H61=    H60+L60 ; LICENSE
30        000654  END=    H61+L61
31
32                .TXTM   1
33                .TXTL   1
34                .TXTX   1
35
36        000651  .TXT    *miscdispsexracepobdobagehtwthaireyeskinmarks*
37        000771  .TXT    *socialdlsdlnflyfbibcicolormakemodelstyleid#lic#*
38
39                .END
```

SYMBOL	VALUE	DEFN	REFERENCES					
H10	000533	13:04	4:26	13:05				
H04	000537	13:05	4:33	13:06				
H05	000542	13:06	5:15	13:07				
H06	000545	13:07	5:26	13:09				
H07	000551	13:09	5:35	13:10				
H11	000553	13:10	6:12	13:11				
H12	000556	13:11	6:19	13:12				
H13	000562	13:12	6:26	13:13				
H14	000565	13:13	6:33	13:14				
H15	000571	13:14	6:40	13:16				
H20	000576	13:16	7:05	13:17				
H21	000601	13:17	7:26	13:18				
H22	000611	13:18	7:33	13:19				
H23	000617	13:19	7:40	13:20				
H24	000612	13:20	6:44	14:21				
H25	000615	13:21	6:47	13:23				
H26	000620	13:23	6:54	14:01				
H01	000625	13:24	7:12	14:05				
H02	000631	13:25	9:12	13:27				
H03	000635	13:26	9:20	13:27				
H04	000642	13:27	9:53	13:28				
H05	000647	13:28	11:12	13:29				
H01	000651	13:29	11:20	14:31				
START	002000	1:11	5:14	5:17	7:17	7:17		
H01	000004	1:25	4:14	4:17	4:21	4:31	5:13	5:17
			6:11	6:17	6:24	6:31	7:13	7:17
			7:24	7:31	7:35	7:38	8:13	8:17
			8:27	9:03	9:10	9:17	9:24	9:31
			10:10	10:17	10:24	10:31	11:13	
H02	000001	1:24						
H03	000014	1:23						
H01	000001	1:22	4:13	4:17	4:21	4:31	5:13	5:17
			6:13	6:17	6:24	6:31	6:34	7:13
			7:24	7:31	7:35	7:38	8:13	8:17
			8:24	8:31	9:03	9:10	9:17	9:24
H02	000002	1:25	5:13	5:17				
H03	000003	1:27	7:14	7:21				
L04	000004	12:10	9:12	11:14	13:02			
L01	000001	12:14	8:27	12:15				
L02	000013	12:15	9:14	12:14				
L03	000005	12:16	10:03	12:05				
L04	000006	12:17	10:10	13:00				
L05	000007	12:18	10:17	13:07				
L01	000008	12:11	9:05	12:01				
L11	000013	12:12	9:12	12:11				
L12	000010	12:13	9:14	12:12				
L13	000011	12:14	9:15	12:13				
L14	000012	12:15	9:16	12:14				
L15	000013	12:16	9:17	12:15				
L16	000014	12:17	9:20	12:16				
L17	000015	12:18	9:21	12:17				
L21	000016	12:19	9:22	12:18				
L22	000017	12:20	9:23	12:19				
L23	000020	12:21	9:24	12:20				
L24	000021	12:22	9:25	12:21				
L25	000022	12:23	9:26	12:22				
L26	000023	12:24	9:27	12:23				
L27	000024	12:25	9:30	12:24				
L30	000025	12:26	9:31	12:25				
L31	000026	12:27	9:32	12:26				
L32	000027	12:28	9:33	12:27				
L33	000030	12:29	9:34	12:28				
L34	000031	12:30	9:35	12:29				
L35	000032	12:31	9:36	12:30				
L36	000033	12:32	9:37	12:31				
L37	000034	12:33	9:40	12:32				
L40	000035	12:34	9:41	12:33				
L41	000036	12:35	9:42	12:34				
L42	000037	12:36	9:43	12:35				
L43	000040	12:37	9:44	12:36				
L44	000041	12:38	9:45	12:37				
L45	000042	12:39	9:46	12:38				
L46	000043	12:40	9:47	12:39				
L47	000044	12:41	9:50	12:40				
L50	000045	12:42	9:51	12:41				
L51	000046	12:43	9:52	12:42				
L52	000047	12:44	9:53	12:43				
L53	000050	12:45	9:54	12:44				
L54	000051	12:46	9:55	12:45				
L55	000052	12:47	9:56	12:46				
L56	000053	12:48	9:57	12:47				
L57	000054	12:49	10:00	12:48				
L60	000055	12:50	10:01	12:49				
L61	000056	12:51	10:02	12:50				
L62	000057	12:52	10:03	12:51				
L63	000060	12:53	10:04	12:52				
L64	000061	12:54	10:05	12:53				
L65	000062	12:55	10:06	12:54				
L66	000063	12:56	10:07	12:55				
L67	000064	12:57	10:10	12:56				
L70	000065	12:58	10:11	12:57				
L71	000066	12:59	10:12	12:58				
L72	000067	13:00	10:13	12:59				
L73	000070	13:01	10:14	13:00				
L74	000071	13:02	10:15	13:01				
L75	000072	13:03	10:16	13:02				
L76	000073	13:04	10:17	13:03				
L77	000074	13:05	10:20	13:04				

LINE	DEF'N	REFERENCES					
3	12:23	4:17	13:23				
4	12:25	4:20	14:24				
5	12:26	9:12	13:25				
6	12:27	9:14	13:26				
7	12:28	9:16	13:27				
8	12:29	9:18	13:28				
9	12:30	11:12	13:29				
10	12:31	11:10	13:30				
11	1:35	11:42					
12	1:34	1:37					
13	1:36	1:39					
14	1:38	2:13	11:43				
15	2:13	3:06	4:06				
16	2:14	3:07	4:13				
17	2:15	3:08	4:20				
18	2:16	3:09	4:27				
19	2:17	3:10	4:34				
20	2:20	3:11					
21	2:09	3:12	5:27				
22	2:11	3:15	6:06				
23	2:12	3:16	6:13				
24	2:13	3:17	6:20				
25	2:14	3:18	6:27				
26	2:15	3:19	6:34				
27	2:16	3:21	8:06				
28	2:18	3:22					
29	2:19	3:23	7:27				
30	2:20	3:24	7:34				
31	2:21	3:25	7:41				
32	2:22	3:26	6:41				
33	2:23	3:27	6:48				
34	2:25	3:31	3:32	3:33	8:13	8:18	8:23
35	2:26	3:35	8:20				
36	2:30	3:40					
37	2:31	3:41	3:13				
38	2:32	3:42	4:0				
39	2:33	3:43	4:27				
40	2:34	3:44	4:34				
41	2:36	3:47	11:13				
42	2:37	3:48	11:19				
43	2:38	3:49	11:13				
44	2:39	3:50	11:27				
45	2:40	3:51	11:27				
46	2:41	3:54	11:19				
47	2:42	3:56	3:57	3:58	11:20	11:25	11:30
48	2:43	4:17	4:24	4:31	5:03	5:24	6:10	6:17
49	1:37	5:21	5:31	6:38	6:45	7:24	7:31	7:38
		8:13	8:15	8:20	8:27	9:10	9:17	9:24
		9:31	11:6	10:10	10:17	10:24	11:17	11:22
		11:27	11:34					
		11:41	11:48					
50	3:5	3:41						
51	3:6	3:30						
52	3:12	3:13						
53	3:17	3:21						
54	3:24	3:24						

SYMBOL	VALUE	DEF'N	REFERENCES
.4.	000544	3:33	3:34
.5.	000547	3:35	3:39
.6.	000572	3:44	3:46
.7.	000632	3:51	3:53
.8.	000642	3:54	3:55
.9.	001032	3:58	

FLAGGED LINES: NONE

TIME UTILITY (TIME.)

The time utility allows the operator to either set the internal system time or to see what it is. The message "TIME" is displayed, at which point the operator may type in the time in four digits such as 1312 for 1:12 P.M. The utility will reset the internal time words and display the new time. If the operator does not type in a new time, the utility merely displays the current time.

```
PAGE  1        A I D S     T I M E     U T I L I T Y
 2/26/74
    1                    .HEAD  A I D S     T I M E     U T I L I T Y
    2
    3                    .TITLE TIME.
    4
    5          0000060   HOURS=  60      ; DIGOS HOURS WORD
    6          0000061   MINS=   61      ; DIGOS MINUTES WORD
    7          0000062   SECS=   62      ; DIGOS SECONDS WORD
    8
    9          0000141   .DIV=   141     ; AIDS DIVIDE
   10          0000142   .CVA=   142     ; ASCII CONVERSION
   11          0000143   .CVB=   143     ; BINARY CONVERSION
   12          0000144   TYPE=   144     ; AIDS TYPE BYTE
   13          0000145   ASK=    145     ; AIDS DATA INPUT
   14
   15          0000005   ZAP=    6       ; DISPLAY ENTRY
   16
   17          0000043   D5=     P3
   18          0000044   D4=     P4
   19          0000045   D3=     P5      ; DECIMAL DIGITS
   20          0000046   D2=     P6
   21          0000047   D1=     P7
   22
   23          0000110   TI=     110     ; TIME INPUT AREA
   24
   25          0000051   RT=     P1      ; RETURN ADDRESS
   26
   27                    .CODE
   28
   29 0000000'205051    TIME.:  STA 3   RT,2    ; SAVE RETURN
   30 0000001'020461            LDA 3   .4
   31 0000002'024401            LDA 1   .DI     ; DATA OBJECT
   32 0000003'117011            ADD     2,1     ; [illegible]
   33 0000004'151000            JSR     @ASK    ; GET DATA
   34 0000005'010431            JMP     @ZAP    ; TELL ZAP
   35
   36 0000006'021111            LDA 0   TI+1,2
   37 0000007'041017            STA 0   D1,2    ; DIGIT 1
   38 0000010'151000            JSR     .DIV
   39 0000011'041016            STA 0   D2,2    ; AND 2
   40
   41 0000012'021111            LDA 0   TI,2
   42 0000013'041015            STA 0   D3,2    ; DIGIT 3
   43 0000014'151000            JSR     .DIV
   44 0000015'041014            STA 0   D4,2    ; AND 4
   45
   46 0000016'112441            SUB     0,0     ; NO DIGIT 5
   47 0000017'041013            STA 0   D5,2
   48 0000020'000062            JMP     @SECS   ; CLEAR SECONDS
   49 0000021'000103            JMP     .OK?    ; [illegible]
   50
   51 0000022'020445            LDA 0   .100.
   52 0000023'151000            JSR     .DIV    ; DIVIDE BY 100.
   53 0000024'020444            LDA 3   .24.    ; [illegible]
   54 0000025'113112            SUB     1,3     ; IF [illegible]
   55 0000026'000407            JMP     .OK?    ; [illegible]
   56 0000027'030436            LDA 3   .60.    ; [illegible]
   57 0000030'111112            SUB     .3      ; [illegible]
   58 0000031'000404            JMP     .OK?    ; [illegible]
```

```
                    A I D S     T I M E     U T I L I T Y
```

```
                    LDA 1   HOURS   ; GET HOURS
                    LDA 0   .+1     ; AND MINUTES
                    JMP     @RET,2  ; RETURN

                    LDA 1   [63     ; TYPE "?"
                    LDA 2   .+1
                    LDA 3   RET,2   ; RETURN ADDRESS
                    JMP     @ZAP,2  ; EXIT THRU ZAP

                    LDA 1   HOURS   ; GET HOURS
                    JSR     .+XXX   ; MAKE ASCII
                    STA 1   DIG1,2  ; DIGIT 1
                    ADD     0,1
                    LDA 1   HOURS   ; DIGIT 2
                    JSR     .+XXX
                    STA 1   .+1     ; FLAGS
                    JMP     @OUT,2  ; OUTPUT

                    LDA 1   MINS    ; GET MINUTES
                    JSR     @MKA    ; MAKE ASCII
                    STA 1   DIG1,2  ; DIGIT 1
                    MOVZR   0,1
                    STA 1   DIG2,2  ; DIGIT 2
                    LDA 1   .+1     ; FLAGS
                    JMP     @OUT,2  ; OUTPUT

                    JMP     @RET,2  ; RETURN

                    .TXT    <4>
                    .BYTE   "
                    .BYTE   "
                    .BYTE   "
                    .BYTE   "
                    .BYTE   "
                    .BYTE   "

                    .TXT    |

                    .BYTE   7*"*"+"?

                    .BYTE   "TIME.
```


SYMBOL	VALUE	DEFN	REFERENCES		
ASK	000105	1:13	1:33		
D1	000147	1:21	1:37	2:12	2:21
D2	000146	1:20	1:39	2:14	2:23
[illegible]	[illegible]	1:19	1:42		
[illegible]	[illegible]	1:18	1:41		
[illegible]	[illegible]	1:17	1:47		
[illegible]	[illegible]	2:17	1:44		
[illegible]	[illegible]	2:37	2:45		
[illegible]	[illegible]	2:35	1:55	1:58	
[illegible]	[illegible]	1:35	2:51	2:14	
[illegible]	[illegible]	1:36	2:12	2:19	
[illegible]	[illegible]	1:26	1:24	2:03	2:27 2:28
[illegible]	[illegible]	1:27	1:13		
[illegible]	[illegible]	1:23	1:36	1:41	2:31
[illegible]	[illegible]	1:22	2:41		
[illegible]	[illegible]	1:12			
[illegible]	[illegible]	1:15	2:43	2:17	2:26
[illegible]	[illegible]	2:35	1:51		
[illegible]	[illegible]	2:34	2:46	2:16	2:25
[illegible]	[illegible]	2:32	1:53		
[illegible]	[illegible]	2:31	1:57		
[illegible]	[illegible]	2:33	1:56		
[illegible]	[illegible]	1:11	2:11	2:23	
[illegible]	[illegible]	1:11	1:49		
[illegible]	[illegible]	1:09	1:52		
[illegible]	[illegible]	2:31	1:51		

FLAGGED LINES: NONE

TIME ZONE UTILITY ("ZONE.)

The time zone utility allows the operator to either set the time zone word or to see what it is. The message "%ZONE" is displayed, at which point the operator may type in the time zone. The utility will reset the time zone word and display the new zone. If the operator does not type in a new zone, the utility merely displays the current zone.


```
                        .HEAD   A I D S    T I M E - Z O N E    U T I L I T Y

                        .TITLE  ZONE.

000104          TYPE=   104     ; AIDS TEXT TYPER
000105          ASK=    105     ; AIDS DATA INPUT
000145          ZONE=   145     ; TIME-ZONE WORD
000006          ZAP=    6       ; TIT DISPLAY ENTRY
000110          TZ=     110     ; DATA INPUT AREA
000051          PT=     51      ; INPUT/OUTPUT ASCII

                        .TWOSEG

000051  ZONE.:  STA 3   PT,2    ; SAVE RETURN
                LDA 1   .3
                LDA 1   .TZ     ; DATA OFFSET
                ADD     2,1     ; ADD TO BUFFER
                LDA     ,1      ; GET CHAR
                JSR             ; GET NUM
                STA 1   TZ,2    ; SET ZONE
                STA 1   ZONE    ; SET ZONE
                JMP     @PT,2   ; RETURN

000117  %ZONO:  LDA 1   ZONE    ; TIME ZONE
                LDA 0   .10
                JSR     ZAP,2   ; DISPLAY IT
                LDA 1   ..T     ; LITERAL "T"
                LDA 0   .110
                JSR     @ZAP,2  ; TACK IT ON
                JMP     @PT,2   ; RETURN

000003  .3:     3
000104  ..T:    "T"
000110  .TZ:    TZ
000111  .110:   110
000131  .130:   130

                .TXTM   1

                .END    ZONE.
```

TITLE	DEFN	REFERENCES			
[illegible]	[illegible]	1:21			
[illegible]	[illegible]	1:22			
[illegible]	[illegible]	1:17	1:26	1:34	
[illegible]	[illegible]				
[illegible]	[illegible]	1:24	1:38		
[illegible]	[illegible]	1:31	1:33		
[illegible]	[illegible]	1:25	1:28		
[illegible]	[illegible]	1:45			
[illegible]	[illegible]	1:32			
[illegible]	[illegible]	1:29			
[illegible]	[illegible]	[illegible]			
[illegible]	[illegible]	[illegible]			
[illegible]	[illegible]	[illegible]			

[illegible] LINES: [illegible]

CENTER UTILITY (CITY.)

The center utility allows the operator to either set the control center name word for his terminal or to see what it is set to. The message "CENTER" is displayed, at which point the operator may type in a center name (two ASCII characters). If the center name is valid, the control center name in the terminal information table is reset and the new center name is displayed. If the operator does not type in a new center name, the utility merely displays the current control center name.


```
                        .HEAD   A I D S    C E N T E R    U T I L I T Y
                        .TITLE  CITY.
                        ;       AIDS TABLE ASSIGNMENTS
0001Ø4          TYPE=   1Ø4     ; AIDS TEXT TYPER
0001Ø5          ASK=    1Ø5     ; AIDS DATA INPUT
000130          .CCN=   13Ø     ; CENTER CONTROL
                        ;       TIT WORD ASSIGNMENTS
000006          ZAP=    6       ; DISPLAY OUT IF ENTRY
000020          CCN=    2Ø      ; CONTROL CENTER NAME
000044          CN=     P0      ; DEFAULT CALL INPUT
000051          RT=     R1      ; RETURN ADDRESS
                        .PROC
                CITY.:  STA 3   RT,2    ; SAVE RETURN
                        LDA Ø   .2
                        LDA 1   .CN     ; DATA OFFSET
                        ADD     2,1     ; MAKE A BYTE
                        JSR     @ASK    ; GET CENTER
                        JMP     ECHO    ; TELL ME
                        LDA 1   2,2     ; INPUT DATA
                        STA Ø   CCN,2   ; SET INTO TIT
                        JSR     @.CCN   ; VERIFY IT
                        JMP     ERROR   ; TYPE ERROR
                        JMP     @RT,2   ; AND RETURN
                ECHO:   LDA 1   CCN,2   ; OPERATOR WANTS
                        SKIP
                ERROR:  LDA 1   ERR     ; TYPE "?"
                        LDA Ø   .13Ø
                        JSR     @ZAP,2  ; DISPLAY IT
                        JMP     @RT,2   ; AND RETURN
                .2:     2
                .CN:    CN
                .13Ø:   13Ø
                .TXTM   1
                ERR:    [illegible]"?
                .END    CITY.
```

[illegible]

[illegible]5	1:[illegible]	1:35		
23	1:17	1:29	1:34	
03′	1:21	1:[illegible]		
43	1:15	1:[illegible]	1:43	
13′	1:[illegible]	1:[illegible]		
24′	1:[illegible]	1:37		
15′	1:37	1:[illegible]		
31	1:[illegible]	1:21	1:32	1:44
104	1:[illegible]			
[illegible]6	1:15	1:[illegible]		
23′	1:[illegible]	1:28		
21′	1:[illegible]	1:22		
3[illegible]	[illegible]	[illegible]		
22′	[illegible]	[illegible]		

[illegible]

OPERATOR UTILITY (STAR.)

The operator utility allows the operator to either set the operator word for his terminal or to see what it is set to. The message "STAR" is displayed, at which point the operator may type in a new operator star number. The operator word in the terminal information table is reset and the new operator star number is displayed. If the operator does not type in a new star number, the utility merely displays the current operator star number.


```
                        .HEAD   A I D S     O P E R A T O R     U T I L I T Y
                        .TITLE  STAR.
                        ;       AIDS TABLE ASSIGNMENTS
000102          .CVA=   102     ; ASCII CONVERSION
000103          .CVB=   103     ; BINARY CONVERSION
000104          TYPE=   104     ; AIDS TEXT TYPER
000105          ASK=    105     ; AIDS DATA INPUT
                        ;       TIT WORD ASSIGNMENTS
000006          ZAP=    6       ; DISPLAY COMMAND ENTRY
000021          OPR=    21      ; OPERATOR NUMBER
000043          D5=     P3
000044          D4=     P4
000045          D3=     P5      ; DECIMAL DIGITS
000046          D2=     P6
000047          D1=     P7
000110          ON=     110     ; DATA INPUT AREA
000051          RT=     R1      ; RETURN ADDRESS
                        .NREL
000000'055051   STAR.:  STA 3   RT,2    ; SAVE RETURN
000001'046104           JSR     @TYPE
000002'000051'          MSG             ; TYPE "STAR"
000003'020443           LDA 0   .4
000004'024444           LDA 1   .ON     ; [illegible]
000005'147000           ADD     2,1     ; [illegible]
000006'056105           JSR     @ASK    ; [illegible]
000007'000416           JMP     ECHO    ; [illegible]
000010'021111           LDA 0   ON+1,2
000011'041047           STA 0   D1,2    ; DIGIT 1
000012'101300           MOVS    0,0
000013'041046           STA 0   D2,2    ; AND 2
000014'021110           LDA 0   ON,2
000015'041045           STA 0   D3,2    ; DIGIT 3
000016'101300           MOVS    0,0
000017'041044           STA 0   D4,2    ; AND 4
000020'102400           SUB     0,1
000021'041043           STA 0   D5,2    ; NO DIGIT 5
000022'046103           JSR     @.CVB   ; [illegible]
000023'[illegible]      STA 1   OPR,2   ; [illegible]
000024'[illegible]      JMP     @RT,2   ; [illegible]
```

```
                ECHO:   LDA 1   OPR.2   ; GET OPERATOR
                        JSR     .CVA    ; MAKE ASCII

                        LDA 1   D4,2    ; DIGIT 4
                        JSR     SEND
                        LDA 1   D3,2    ; DIGIT 3
                        JSR     SEND
                        LDA 1   D2,2    ; DIGIT 2
                        JSR     SEND
                        LDA 1   D1,2    ; DIGIT 1
                        JSR     SEND

                        JMP     @RT,2   ; RETURN

                SEND:   LDA 0   D5,2
                        SNE     0,1
                        JMP     @,3
                        DSZ     D5,2
                        LDA 0   .110
                        JMP     @ZAP,2

                .4:     4
                .110:   110
                .0%:    0"1

                .TXTM   1

                        .ASCI   "GIA?*

                        .END    STAR.
```



```
SYMBOL  VALUE   DEF'N    REFERENCES

ASK     000105  1:10    1:35
D1      000047  1:21    1:
D2      000046  1:20    1:
D3      000045  1:19    1:
D4      000044  1:18    1:
D5      000043  1:17    1:49    2:15    2:18
ECHO    000025' 2:01    1:
MSG     000051' 2:28    1:
        000110' 1:23    1:38    1:43    2:24
OPR     000021  1:15    1:51    2:
        000001  1:25    1:29    1:52    2:13
SEND    000040' 2:15    2:05    2:07    2:09    2:11
STAR.   000000' 1:29    2:
TYPE    000104  1:09    1:
ZAP     000036  1:14    2:
.110    000041' 2:23    2:
.4      000046' 2:22    1:
.CVA    000102  1:07    2:
.CVS    000103  1:08    1:
.ON     000050' 2:24    1:

        FLAGGED LINES:  NONE
```

TICKET CLEAR UTILITY (DROP.)

The ticket clear utility allows the operator to cancel a current ticket. The utility calls the ticket status access routine to delete the current ticket's entry in the ticket status table. The utility will complain if no such entry exists in the ticket status table.


```
                        .HEAD  A I D S   T I C K E T   C L E A R
                        .TITLE DROP.
                TS2=    125     ; TICKET STATUS DELETE
                ZAP=    6       ; DISPLAY COMMAND ENTRY
                COM=    20      ; CONTROL CHARACTER
                TKT=    21      ; ADDRESS TICKET NUMBER
                RT=     R1      ; RETURN ADDRESS
                        .TEXT
DROP.:          SFA 3   RT,2    ; SAVE RETURN
                LDA 0   COM,2   ; CONTROL CHAR
                LDA 1   TKT,2   ; TICKET
                LDB     1,TS2   ; DELETE
                SKIP            ; NOT IN
                J       RT,2    ;
                LDA 1   ERR     ; TYPE "?"
                LDA 3   .13Ø
                LDA 3   RT,2    ; RETURN ADDRESS
                J       ZAP,2   ; EXIT
.13 :           13Ø
ERR:            ?<#>?
                .END    DROP.
```

[illegible] TICKET CLEAR UTILITY

[illegible] DATE REFERENCES

1:11 1:15
1:15 1:31
1:[illegible] 1:22
1:11 1:15 1:21 1:24
1:[illegible] 1:17
1:[illegible] 1:[illegible]
1:47 1:23
1:[illegible] 1:[illegible]

[illegible]

EXIT UTILITY (EXIT.)

The exit utility is used to exit from DINOS and return to DDOS. This utility may only be initiated from the system console device, whose terminal identifier is 4. If this utility is initiated from any other terminal, the utility sends a bell and a question mark to the terminal.


```
                        .HEAD   A I D S     E X I T     U T I L I T Y
                        .TITLE  EXIT.
                ØØØØØØ  ID=     Ø
                ØØØØØ6  ZAP=    6
                        .ENT    EXIT.
                        .NREL
ØØØØØØ'Ø21ØØØ   EXIT.:  LDA Ø    ID.2
ØØØØ1'Ø24406            LDA 1   ./
ØØØØ2'1Ø6513            SGT     ?.1
ØØØØ3'ØØØØ42            EXIT
ØØØØ4'Ø24405            LDA 1   END
ØØØØ5'Ø2Ø4Ø3            LDA Ø   .13Ø
ØØØØ6'ØØ3ØØ6            JMP     @ZAP,2
ØØØØ7'ØØØØØ7    .7:     7
ØØØ1Ø'ØØØ13Ø    .13Ø:   13Ø
ØØØ11'ØØ3477    ERR:    7[illegible]?
ØØØØØ1'         .END    EXIT.
```

[illegible] EXIT UTILITY

[illegible] REFERENCES

[illegible] 1:22 1:15
[illegible] 1:11 1:23 1:24
[illegible] 1:11
[illegible] 1:13
[illegible] 1:17
[illegible] 1:12

[illegible]

TERMINAL PRINT UTILITY (ROLL.)

The terminal print utility displays a listing of all terminals which are active, giving terminal number, operator number and control center name.


```
                        .HEAD   A I D S   T E R M I N A L   P R I
                        .TITLE  ROLL.
                
000101          .DIV=   101     ; DIVIDE ROUTINE
000102          .CVA=   102     ; CONVERT TO ASCII
000104          TYPE=   104     ; AIDS TEXT TYPER
000155          CTIB=   155     ; FIRST TIP ADDRESS
                
000000          ID=     0       ; TERMINAL TYPE
000002          LINK=   2       ; CALL/RET LINK
000003          SAV=    3       ; TEMPORARY STORAGE
000006          ZAP=    6       ; DIC ENTRY PT
000014          EDIT=   14      ; ACCEPT EDITING
000015          DATA=   15      ; TIP DATA BLOCK
000021          CCH=    21      ; CC NAME
000021          CNT=    21      ; CHAR COUNT
000021          PNT=    21      ; TEXT PNTR
000025          OPR=    25      ; OPR TYPE MSGS.
000027          PRI=    27      ; PR QUEUE LEVEL
                
000044          D4=     44
000045          D3=     45      ; DECIMAL
000046          D2=     46      ; DIGITS
000047          D1=     47
                
000051          RT=     51      ; RETURN ADDRESS
                
000000          USE=    0       ; BLOCK USE COUNT
                
                        .PAGE

000000 054051   ROLL.:  STA 3   RT,2     ; SAVE RETURN
000001 102214   ROLL1:  LDA 0   TKT,2    ; CURRENT TICKET
000002 152014           MOV#    0,0 SZR ; IF NO TKT,ERR0.
000003 043041           STA 1   CCH,2    ; PROMPTS OLD CC
000004 034015           LDA 3   DATA,2   ; DATA BLOCK
000005 175014           SUB#    3,3 SZR ; IF IDT ERR0.
000006 054046           STA 3   USE,3    ; PR JOB ASSIGN
000007 000441           JSR              ; IF CASE OF ERR
                
000010 125400           SUB     1,1
000011 040021           STA 1   TKT,2    ; CLEAR TICKET
000012 040015           STA 1   DATA,2   ; AND DATA PTR
000013 040014           STA 1   EDIT,2   ; CLEAR EDIT FLAG
000014 040027           STA 1   PRI,2    ; AND PR QUEUE
000015 125441           SUB#    1,1 SZR  ; SKIP IF ERROR
000016 000435           JMP     .ERR0.   ; ERROR EXIT
                
000017 177433 000155            LDA 2   @CTIB   ; GET TIP ADDR
000021 021000           LOOP:   SUB#    3,3 SNR ; END OF TIPS?

                        ; ASSIGN TIP TO

000021 025421                   LDA 1   @3,3    ; GET TIP ADDR
000022 125000                   SUB     1,1     ; CLEAR AC1
000023 000441                   JSR             ; ERROR.
000024 000435                   JMP     .ERR0.  ; ERROR EXIT
```

```
                        MOVA    R4,2    ; DIGIT 4
                        SWAB    0,1     ; MOVE TO TOP
                        MOVA    R3,2    ; DIGIT 3
                        ADD     1,1     ; COMBINE
                        MOV     1,@PP1  ; AND SAVE

                        MOVA    R2,2    ; DIGIT 2
                        SWAB    0,1     ; MOVE TO TOP
                        MOVA    R1,2    ; DIGIT 1
                        ADD     1,1     ; COMBINE
                        MOV     1,@PP2  ; AND SAVE

        ; GET TERMINAL TABLE

                        MOV     #TTBL,0 ; TERMINAL TABLE
                        MOV     #1,1    ; GET TERMINAL ID
                        MOVA    R0,R    ; RADIX IS 8.
                        MOV     #.+10,0 ; GET TO DIGITS
                        MOV     #"00,@0 ; TWO ASCII ZEROES
                        ADD     #3,0
                        MOV     1,1     ; COMBINE
                        MOV     1,@PP3  ; AND SAVE

        ; OPERATOR OPTION

                        MOV     #TTBL,0 ; TERMINAL TABLE
                        MOV     #2,1    ; OPERATOR NUMBER
                        MOV     #OUTPUT,2 ; SET UP FOR OUTPUT
                        MOV     #.+10,3 ; PRINT IT OUT
                        JSR     R5,@#4

                        MOV     #TTBL,0 ; TERMINAL TABLE
                        MOV     #3,1    ; ROLLOUT/IN LINK
                        MOV     #1,2    ; ATTRIBUTE BYTE
                        MOV     #.+8,3  ; GO CHECK TO SEE
                        MOV     1,LOOP  ; THEN GO DO IT
                        RTS     PC      ; ELSE RETURN

2$:     .BYTE   1,0
3$:     .BYTE   "00*4+1 ; TWO ASCII ZEROES
4$:     .BYTE   1,177760 ; SCOPE ON CLEAR FLAGS
        .BYTE   1
        .BYTE   1
        .BYTE   1
PP1:    .BYTE   15*4+10+12
PP2:    .BYTE   *  TERMINAL  *
        .BYTE   0
PP3:    .BYTE   *  OPERATOR  *
        .BYTE   0
        .BYTE   0
PP4:    .BYTE   *  CENTER  *
        .BYTE   0

        .END    ROLL.
```



```
SYMBOL  VALUE   DEFN    REFERENCES

CON     000001  1:16     2:
PP1     000112  2:63     2:29
PP2     000155  1:     1:     2:35
1       000047  1:25     2:1
2       000045  1:     2:
3       000043  1:23     2:
4       000044  1:22     2:2
APR     000016  1:16     1:     1:44
LP      000001  1:1      1:
        000007  1:21     1:
        000001  1:1      2:17
        000000  1:11     2:1
        000024  1:61     2:
        000020  2:45     2:31
        000036  1:12     1:
        000001  1:17     1:
        000134  2:     2:
        000112  2:     2:
        000007  1:33     2:
        000001  1:27     1:     2:38
        000003  1:     1:     2:16    2:27    2:33
        000141  2:     2:
        000144  2:63     1:
        000024  1:     1:     1:43
        000004  1:     2:
        000006  1:     1:
        000016  1:13     1:
        000034  2:12     1:
        000012  2:     2:
        000034  2:     2:
        000010  1:     1:
        000011  1:     2:

        FLAGGED LINES:  NONE
```

TICKET DIGEST UTILITY (DIGT.)

The ticket digest utility displays one line of information for each ticket present on disk on the terminal display device. It displays in the area normally used for status displays. If the terminal has a control center name of zero (binary), all tickets are listed, otherwise only those tickets belonging to the terminal's control center are listed. Tickets are displayed starting with the most recent and progressing to the oldest ticket on disk. The starting ticket number is extracted from the system vector table. The utility keeps an address plugged into the end control word so that when the screen area is filled, the process stops but can be made to continue by striking the END key.

12/23/74

```
                        .HEAD   A I D S    T I C K E T    D I G E S
                        ;       TERMINAL INFORMATION BLK OFFSETS
000020          CCN=    20      ; CONTROL CENTER NAME
000015          DATA=   15      ; DATA BLOCK ADDRESS
000027          END=    27      ; END CONTROL WORD
000014          FORM=   14      ; FORMAT BLOCK ADDRESS
000013          HOME=   13      ; CURSOR REST POSITION
000006          ZAP=    6       ; DISPLAY OUTPUT VECTOR

000047          D1=     P7      ; LOW ORDER DIGIT
000046          D2=     P6
000045          D3=     P5
000044          D4=     P4
000043          D5=     P3      ; HIGH ORDER DIGIT

                ;       AIDS SYSTEM VECTORS

000141          SAD=    141     ; SCREEN ADDRESS FOR STATUS
000142          MAD=    142     ; SCREEN ADDRESS FOR MESSAGE
000134          TYPE=   134     ; TEXT STRING TYPE
000133          .CCN=   133     ; LOCATE CENTER NAME
000110          .DUMP=  110     ; DISPLAY IN FORMAT
000102          .CVA=   102     ; ASCII CONVERSION ROUTINE
000157          NEXT=   157     ; NEXT TICKET NUMBER

000076          PD=     76      ; OFFSET TO DEPT FIELD

                .TITL   DIGT.
                .ENT

                LIST.:  STA 3   TEMP    ; SAVE RETURN ADDRESS
000001021020            LDA 0   CCN.2   ; GET TICKET CCN
                        LDA 1   ZERO    ; ZERO CCN MEANS ALL TICKETS
                        SUB     1,1     ; SET CCN=-1
                        LDA 0   @0,3    ; GET TICKET NUMBER
                        LDA 1   1,3     ; GET CCN OF TICKET
                        SUB     1,0,SNR ; SAME CCN?
                        JMP     LOOP    ; YES, LIST IT
                        JMP     @TEMP   ; NO, RETURN
                LOOP:   STA 1   CCN     ; SAVE CCN
                        LDA 1   MASK    ; GET BIT MASK FOR TICKET NUMBER
                        ADD     1,1     ; SHIFT MASK
                        STA 1   MASK    ; SAVE MASK
                        LDA 1   @MASK   ; GET MASK
000153004005            STA 1   DEL     ; SET DELETE
000014021014            LDA 0   FORM,2  ; GET INITIAL FORMAT BLOCK ADDR
000015040561            STA 0   IFORM   ; SAVE IT
000016021015            LDA 0   DATA,2  ; GET INITIAL DATA BLOCK ADDR
                        STA 1   IDATA   ; SAVE IT
                        XOR     1,1

                DOIT:   STA 3   RM,2    ; SAVE RETURN ADDRESS
                        LDA 1   MAD     ; GET MESSAGE AREA ADDRESS
000023024141            LDA 1   SAD     ; AND STATUS AREA ADDRESS
000024122700            SUBS    1,0     ; SET UP COUNT
000025040565            STA 0   COUNT   ; FOR MOVE

000026023541            LDA 0   ERASE   ; SET FLAGS TO POSITION
000027024142            LDA 1   MAD     ; GET POSITION OF MESSAGE
```

```
                LDA 3   MPOS    ; AND OFFSET
                ADD     3,1     ; COMPUTE ADDRESS
                JSR     @ZAP,2  ; ERASE THE MESSAGE

TLOOP:          LDA 1   TKT     ; PICK UP TICKET NUMBER
                JSR     @.CVA   ; CONVERT TO ASCII

                LDA 0   D5,2    ; DIGIT 5
                MOVS    0,1     ; TO TOP BYTE
                LDA 0   D4,2    ; DIGIT 4
                ADD     1,0     ; COMBINE
                STA 0   FN1     ; AND SAVE

                LDA 0   D3,2    ; DIGIT 3
                MOVS    0,1     ; TO TOP BYTE
                LDA 0   D2,2    ; DIGIT 2
                ADD     1,0     ; COMBINE
                STA 0   FN2     ; AND SAVE

                LDA 0   D1,2    ; DIGIT 1
                MOVS    0,0     ; TO TOP BYTE
                STA 0   FN3     ; AND SAVE

        ;       SEARCH FOR THE TICKET

                SUBZL   0,0     ; CREATE A +1
                DEQ             ; GRAB THE DISK
                LDA 3   DDOS    ; GET SYSTEM ADDRESS
                LDA 2   .FN     ; POINT TO FILE NAME
                JSR     @D.SRC,3        ; SEARCH FOR TICKET
                JMP     EXYZ    ; TICKET NOT PRESENT

        ;       PRIME INPUT BUFFER

                LDA 0   F.ID,2  ; GET FILE IDENTIFIER
                STA 0   D.SP3,3 ; SET UP PARAMS FOR PRIME
                LDA 0   F.FST,2 ; HIGH ORDER DISK ADDRESS
                LDA 1   F.BFP,2 ; LOW ORDER DISK ADDRESS
                LDA 2   .377    ; HALF WORD MASK
                MOVZR   0,0     ; SHIFT HIGH ORDER ADR, THEN
                ADCL    2,1     ; LOW ORDER, WITH CARRY
                LDA 2   PUF     ; BUFFER PREFIX ADDRESS
                JSR     D.PRM,3 ; PRIME INPUT BUFFER

        ;       SET DATA BLOCK POINTERS

                LDA 2   2       ; RESTORE BASE
                SUBZL   0,0     ; CREATE A +1
                DEQ             ; RELEASE THE DISK
                LDA 3   PUF     ; GET BUFFER PREFIX ADDRESS
                LDA 0   DEPT    ; GET THIS DEPT. NAME
                MOV     0,0 SNR ; ZERO?
                JMP     CCNOK   ; YES, ALLOW ANY CCN
                LDA 1   PD,3    ; GET TICKET'S DEPT
                SNE     0,1     ; SAME?
                JMP     CCNOK   ; YES, OUTPUT

TRAP:           DSZ     TKT     ; MOVE TO NEXT TICKET
```



```
                JMP     TLOOP   ; TRY NEW TICKET
                JMP     EXYT    ; TKT# = 0, TIME TO QUIT

CCNOK:          DSZ     CCNUM   ; BUMP LIST POINTER
                JMP     SEND    ; IT HITS, FINE

                SUB     0,0     ; CLEAR FLAGS FOR POSITION ZAP
                LDA 1   MAD     ; GET
                LDA 3   MPOS    ; GET
                ADD     3,1     ; COMPUTE
                JSR     @ZAP,2  ; POSITION
                JSR     @TYPE   ; TYPE
                MESS            ; MESSAGE

RSET:           LDA 1   IFORM   ; GET INITIAL FORMAT BLOCK ADDRESS
                STA 1   FORM,2  ; RESTORE INITIAL
                LDA 1   IDATA   ; GET INITIAL DATA BLOCK ADDRESS
                STA 1   DATA,2  ; RESTORE INITIAL

                LDA 1   .DOIT   ; GET
                STA 1   END,2   ; PLUG
                JMP     @R0,2   ; RETURN

SEND:           STA 3   DATA,2  ; SET DATA BLOCK ADDRESS
                LDA 1   .FMT    ; PICK UP FORMAT ADDRESS
                STA 1   FORM,2  ; SET FORMAT BLOCK ADDRESS

                LDA 1   COUNT   ; PICK UP LINE COUNTER
                INCS    1,1     ; CONVERT TO A LINE ADDRESS
                LDA 0   MAD     ; GET MESSAGE SCREEN ADDRESS
                SUB     1,0     ; COMPUTE SCREEN ADDRESS
                STA 0   HOME,2  ; SET NEW CURSOR POSITION
                LDA 3   .DUMP   ; GET ADDRESS OF OUTPUT ROUTINE
                JSR     1,3     ; OUTPUT
                JMP     TRAP    ; GO GET NEXT TICKET

EXYZ:           SUBZL   0,0     ; CREATE A +1
                DEQ             ; RELEASE THE DISK

EXYT:           DSZ     COUNT   ; BUMP COUNTER
                JMP     CLEAR   ; MORE LINES TO CLEAR

                LDA 1   NEXT    ; GET STARTING TICKET NO.
                STA 1   TKT     ; RESET TICKET NUMBER
                JMP     RSET    ; RESET TIT WORDS AND GET

CLEAR:          LDA 0   COUNT   ; GET LINE COUNTER
                INCS    0,0     ; ADD 1
                LDA 1   MAD     ; GET
                SUB     .1      ; COMPUTE
                LDA 0   ERASE   ; ERASE
                JSR     @ZAP,2  ; ZAP
                JMP     EXYT    ; DONE?

ERROR:          LDA 1   ERR     ; GET
                LDA 0   .130    ; TO
                LDA 3   RTRN    ; RETURN
                JMP     @ZAP,2  ; ZAP
```

```
                        ;RETURN ADDRESS
                FMT     ;OUTPUT FORMAT
                        ;BELL AND "?"
                        ;FLAGS FOR TWO BYTES
                        ;FLAGS TO ERASE A LINE
                        ;TERMINAL CHARACTER
                .+1     ;FILE NAME POINTER
                        ;1ST WORD
                        ;2ND WORD
                        ;3RD WORD
                        ;FILE WORD MASK
                        ;INITIAL FORMAT BLOCK ADDRESS
                        ;INITIAL DATA BLOCK ADDRESS
                        ;CURRENT NUMBER
                        ;TWO-CHAR. NAME
                        ;CURRENT / OFFSET
                        ;SECTOR COUNTDOWN
                .+1
                .TXT    "
                        ;TICKET NUMBER
                        ;INCIDENT CODE
                        ;INCIDENT LOCATION
```



```
                        ;DATE
                .BUF:   .+1
                        .BLK    B.LEN
                .END    LIST.
```

A I D S T I C K E T D I G E S T U T I L I T Y

[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
[illegible]	5:23	2:42	2:51			
[illegible]	1:[illegible]	1:34				
[illegible]	3:[illegible]	2:53	2:56			
[illegible]	3:47	3:41				
[illegible]	4:17	1:[illegible]	3:[illegible]	3:28	3:40	3:47
[illegible]	1:12	2:[illegible]				
[illegible]	1:13	2:16				
[illegible]	1:[illegible]	3:[illegible]				
[illegible]	1:[illegible]	2:14				
[illegible]	1:[illegible]	4:[illegible]				
[illegible]	1:[illegible]	1:47	3:18	3:24		
[illegible]	1:[illegible]	1:[illegible]	4:51			
[illegible]	1:41	4:12				
[illegible]	1:17	3:21				
[illegible]	4:[illegible]	1:37	3:51			
[illegible]	4:[illegible]	3:[illegible]				
[illegible]	2:[illegible]	1:[illegible]	1:43			
[illegible]	2:[illegible]	3:[illegible]	3:53			
[illegible]	1:21	2:31				
[illegible]	1:[illegible]	4:13				
[illegible]	1:[illegible]	2:[illegible]				
[illegible]	1:[illegible]	2:17				
[illegible]	1:[illegible]	2:[illegible]				
[illegible]	1:[illegible]	1:[illegible]	3:16	3:26		
[illegible]	1:[illegible]	3:32				
[illegible]	1:17	1:43	3:17			
[illegible]	4:13	1:[illegible]	3:15			
[illegible]	1:[illegible]	2:[illegible]				
[illegible]	1:21	1:[illegible]	1:[illegible]	3:[illegible]	3:30	3:49
[illegible]	4:[illegible]	2:[illegible]				
[illegible]	1:47	2:[illegible]	3:[illegible]			
[illegible]	1:46	1:41	3:43			
[illegible]	1:[illegible]	2:[illegible]				
[illegible]	3:15	3:[illegible]				
[illegible]	2:[illegible]	1:[illegible]	3:[illegible]			
[illegible]	1:[illegible]	1:[illegible]				
[illegible]	3:[illegible]	3:[illegible]				
[illegible]	4:[illegible]	1:[illegible]	2:[illegible]	2:58	3:44	
[illegible]	2:[illegible]	3:[illegible]				
[illegible]	2:[illegible]	4:[illegible]				
[illegible]	1:[illegible]	3:[illegible]				
[illegible]	1:[illegible]	2:[illegible]	3:11	3:52	3:58	
[illegible]	1:[illegible]	1:[illegible]				
[illegible]	4:[illegible]	3:[illegible]				
[illegible]	1:[illegible]	2:[illegible]				
[illegible]	1:23	1:47				
[illegible]	1:25	3:[illegible]				
[illegible]	4:[illegible]	3:[illegible]				
[illegible]	1:2[illegible]	3:[illegible]				
[illegible]	4:[illegible]	3:[illegible]				
[illegible]	4:[illegible]	1:[illegible]				

DAILY RECAP SUMMARY (RECAP)

The daily recap summary creates a printer file report of all tickets that are more recent than the last time the daily recap summary was run. The starting ticket number is extracted from the control center table and eventually replaced by the finishing ticket number. The printer file is named "XX.RECAP" where "XX" is the control center name, each center having its own recap report.

GENERAL INFORMATION

The structure of the data base is defined in the system vector table, terminal information tables, format blocks and data blocks, status tables and the control center table, also known as the center assigned resource table.

The software falls into several general groups:

INPUT/OUTPUT

TRIO -- Terminal input and output

DZAP -- ADDS CRT display output

PZAP -- G.E. Terminet printer output

DZIP -- Dispatch console input

CZIP -- Control console input

KEYBOARD DISPATCH/FUNCTIONS

AWAIT -- Keyboard data entry

CALL. -- Invoke function

MAKE. -- Event function

UNIT. -- Unit function

FIND. -- Ticket function

STAT. -- Status function

ZERO. -- Clear function

DEBUG. -- Debug function

FILE. -- File function

MEMO. -- Message function

RADIO -- Radio function

PRINT -- Print function

Functions are generally initiated by striking a function key followed by a dispatch to the proper function from the AWAIT routine.

FORMAT DATA MANIPULATION

DUMP. -- Format display

GET. -- Format read

PUT. -- Format write

EDIT. -- Format edit

TRANS -- Data block transformatter

WATCH -- Update monitor

STATUS ROUTINES

USA. -- Unit status access

USO. -- Unit status output

TSA. -- Ticket status access

TSO. -- Ticket status output

INTRINSIC ROUTINES

CCN. -- Control center lookup

ASK. -- Data string input

TYPE. -- Text string output

CVA. -- ASCII conversion

CUB. -- Binary conversion

MPY. -- Multiply subroutine

DIV. -- Divide subroutine

POST. -- Location lookup

CODE. -- Incident lookup

UTILITIES

Utilities may either be invoked by the operator or initiated by an internal call to the invoke function. The normal method of initiation is given below.

MESSAGE FUNCTION UTILITIES

TO. -- Message function utility

UNIT FUNCTION UTILITIES

TEN42 -- Unit off duty

TEN44 -- Unit on break

TEN2Ø -- Change location

TEN86 -- Change name

TEN76 -- Unit assignment

TEN22 -- Disregard assignment

TEN23 -- Unit arrived

TEN24 -- Unit completed

TEN8 -- Unit in service

TEN41 -- Unit on duty

OTHER -- Other ten-codes

TICKET FUNCTION UTILITIES

ADD. -- Append utility

DROP. -- Ticket clear utility

LIST. -- Ticket digest utility

(continued)

(UTILITIES -- continued)

OPERATOR UTILITIES

TIME. -- Time utility

ZONE. -- Time zone utility

CITY. -- Center utility

STAR. -- Operator utility

EXIT. -- Exit utility

ROLL. -- Terminal print utility

REPORT GENERATORS

RECAP -- Daily recap report generator

FORMAT BLOCKS AND DATA BLOCKS

Ticket information is stored in data blocks ith information packed two characters in a word. All access to the data block is under format control through the use of a format block. For each field in the data block, there is a four-word element in the format block. For display purposes, field header information is stored at the end of the format block.

The format elements are as follows:

WORD ∅ flags

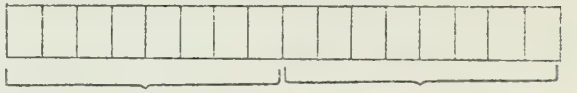
BIT	USE
∅	Last field flag -- indicates last field in format block
1	Advance stop -- EDIT will stop at this field on an advance request
2	Tab stop -- EDIT will stop at this field on a tab request
3	Back stop -- EDIT will stop at this field on a reverse tab request
4	Continuation field -- this field is a continuation of the preceding field
5	Header bright -- display header bright. (Headers are normally displayed dim.)
6	Data dim -- display data dim. (Data is normally displayed bright.)
7	Data term. CR, LF -- terminate data with carriage return and line feed.
8 & 9	Data term. space count -- number of spaces after data (∅, 1, 2 or 3)
10 & 11	Data terminator: ∅ = none, 1 = dot, 2 = colon, 3 = dash

(continued)

BIT	USE
12 & 13	Header terminator space count -- number of spaces after header (∅, 1, 2, or 3)
14 & 15	Header terminator: ∅ = none, 1 = colon, 2 = slash, 3 = dash

Word 1 contains the display position of the first data character. The position of the header must be calculated from the header byte count, the header space count and the header terminator, if any. The display order is: header, header terminator, header space, data, data terminator, data space. The entire header is displayed dim unless flag bit 5 is set. The entire data field is displayed bright unless flag bit 6 is set.

WORD1



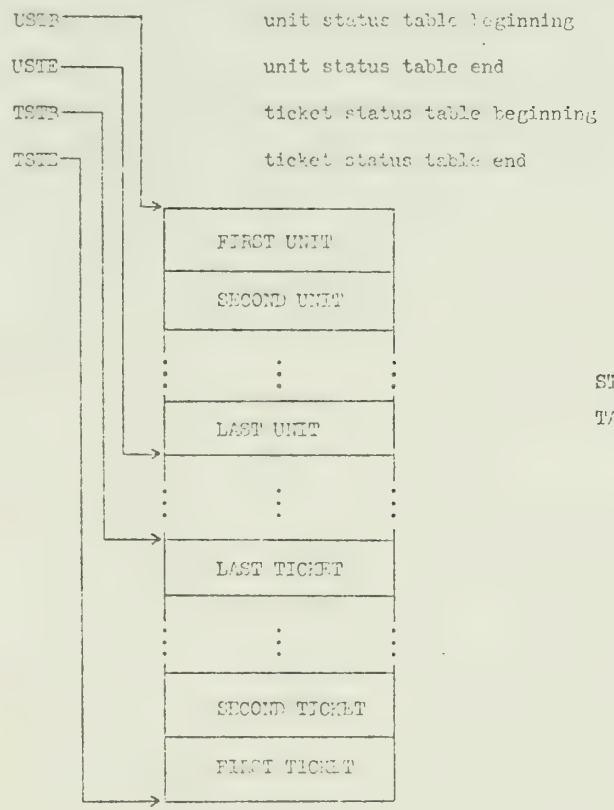
WORD2

WORD3

The first few words of a format block may contain displacements to format elements and are defined at the beginning of the format. See listing of TICKET FORMAT (.FORM) for an example.

STATUS TABLES

Both ticket and unit status information is kept in the status table. The size and location of the status table is defined in the system vector table SVT. Unit status is kept at the top of the table and ticket status is kept at the bottom. Each status entry is 4ø words (octal) long. Four status table pointers are defined in the SVT. They are:



(continued)

The contents of each 4ø word status table entry is as follows:

WORD	CONTENTS
ø	control center name
1	unit number in unit status; ticket number in ticket status
2	ten code
3	time
4	post
5	ticket number in unit status; unit number in ticket status
6	assignment area in unit status; priority in ticket status
7	time limit for watch dog timer
1ø - 23	24 character ASCII string (nature of incident in unit status) (officers' names in ticket status)
24 - 37	24 character ASCII string (location of incident or unit)

TRANSFORMATION INTERLOGIC

The transformatter is used to change the format of a data block. It is given two format blocks and a correlation block which says what field in the source format moves to what field in the destination format. Its five address values are passed in the first five parameter words of the terminal information table as follows:

P0 -- Source format block address

P1 -- Source data block address

P2 -- Destination format block address

P3 -- Destination data block address

P4 -- Correlation block address

BOUND BY
HNM

UNIVERSITY OF ILLINOIS-URBANA
510.84IL63C C001
CAC DOCUMENT$URBANA
148 1974
3 0112 007263863